MATLAB 7.0 实用指南

（下册）

苏金明　王永利　编著

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

本套书基于 MATLAB 的最新版本 7.0 分上下两册详细介绍该软件的使用方法，主要内容包括 MATLAB 7.0 的入门知识、界面设计、编译、接口以及新版本变化较大的图形功能和图像处理、虚拟现实、地图制作等 3 个工具箱。

本书为下册，主要介绍 MATLAB 的图像处理、虚拟现实和地图制作等 3 个工具箱。图像处理部分介绍图像合成，空间变换，邻域和块处理，线性滤波和滤波器设计，基于区域的处理，变换域处理，数学形态学，图像分析，图像增强，图像配准和图像恢复等图像处理技术的实现方法。虚拟现实部分介绍利用 MATLAB 的虚拟现实工具箱创建和浏览虚拟场景并进行交互的方法。地图制作部分介绍地理空间数据、地理空间几何和地图投影等基础知识和实现方法，以及如何利用地图制作工具箱绘制和定制二维、三维地图。

本书内容全面，新颖，适合相关专业的大学生、研究生、科研人员和科技工作者阅读。

图书在版编目（CIP）数据

MATLAB 7.0 实用指南. 下册/苏金明，王永利编著. —北京：电子工业出版社，2004.11
ISBN 7-121-00450-X

Ⅰ. M… Ⅱ. ①苏…②王… Ⅲ. 计算机辅助计算—软件包，MATLAB 7.0 Ⅳ. TP391.75

中国版本图书馆 CIP 数据核字（2004）第 104606 号

责任编辑：龚兰方
印　　刷：北京牛山世兴印刷厂
出版发行：电子工业出版社
　　　　　北京市海淀区万寿路 173 信箱　邮编　100036
经　　销：各地新华书店
开　　本：787×1092　1/16　印张：18　字数：460 千字
印　　次：2004 年 11 月第 1 次印刷
印　　数：5 000 册　　定价：28.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

前　言

近年来，MATLAB 以其强大的矩阵计算和图形可视化功能逐渐为国人所知。很多学校已经开设这方面的课程，很多学生已经开始使用该软件完成论文设计。科学计算软件的使用，极大地提高科研人员的工作效率，可以更快、更准确地完成计算方案的设计，可以在必要的时候用图形图像表示计算结果和描述运行机制。

本书基于 MATLAB 7.0 版本，分上、下两册介绍该软件的使用。相对于以前诸版本，7.0 版本在图形和编译器方面有比较明显的变化，部分工具箱也有一些变化。本书的主要内容可以概括为两个部分，一部分系统介绍 MATLAB 的基础和核心功能，即 MATLAB 总包的功能；另一部分系统介绍新版本变化最大的图形图像功能。上册主要介绍 MATLAB 总包的应用，下册主要介绍几个图形图像方面的工具箱。

第 1～第 16 章为上册的主要内容，系统地介绍 MATLAB 7.0 的基本特点、运行环境、数组、矩阵、数值计算、M 文件、图形用户界面设计、编译、接口，以及二维、三维图形功能。第 1～第 5 章为比较基础的内容，适合于初学者入门；第 6 章介绍最新的编译器 4.0 和 COM 生成器 1.1。编译器 4.0 可以接受对象数据类型，这在以前是不行的。利用 COM 生成器可以将 MATLAB 的 M 文件和 MEX 文件打包成 COM 组件，这些组件又可以用于支持 COM 机制的应用程序，如 VC, VB 等，从而可以实现无缝集成。

MATLAB 7.0 的最大亮点就在于添加了图形的交互创建和编辑功能。这里所说的交互，指的是鼠标交互，即主要通过鼠标的单击和拖拉操作完成图形的绘制和编辑。交互功能的添加，提高了绘图效率和绘图准确性。与此相对应，作为 MATLAB 图形图像和界面基础的句柄图形对象也有了很大的改变。这种改变，体现在对象抽象和对象组织上。

具体来说，图形部分的内容包括二维图形绘制、图形的交互创建和编辑、二维图形的定制、三维模型和场景的创建和变换，以及 MATLAB 提供的一系列科学计算可视化工具等。二维部分，MATLAB 可以绘制条形图、等值线图、向量图等几十种图形，利用图形对象创建函数，还可以实现图形定制；三维部分，可以创建三维网格图、曲面图、流线图、剖面图、等值面图等多种图形。三维程序的开发，是一件富有挑战性但又其乐无穷的事情。在这方面，MATLAB 实际上提供了一个比较高的平台。本书分表面模型和多边形模型两种情况，全面介绍三维模型的创建、着色、光照、材质、透明性、纹理映射和交互操作。

第 17～第 42 章为下册的主要内容，主要介绍 MATLAB 的图像处理、虚拟现实和地图制作工具箱。

第 17～第 30 章介绍图像处理工具箱，内容包括图像合成、空间变换、邻域和块处理、线性滤波和滤波器设计、基于区域的处理、变换域处理、数学形态学、图像分析、图像增强、图像配准和图像恢复等。

第 31～第 35 章介绍虚拟现实工具箱，内容包括虚拟场景的创建、浏览和交互。

第 36～第 42 章介绍地图制作工具箱，例如，地理空间数据、地理空间几何和地图投影等基础知识和实现方法，还介绍如何利用地图制作工具箱绘制和定制二维、三维地图。

目前，虚拟现实在科研方面迅速地向很多专业领域渗透，是当前计算机图形学研究的三大热点之一。传统的实现方法是使用 OpenGL, DirectX3D 等 API 和 VRML,VEGA 等语言，需要使用者具有较多的知识储备。而 MATLAB 的虚拟现实工具箱提供专门的 VRML 编辑器和虚拟场景查看器，可以在不懂 VRML 语言的情况下实现虚拟场景的创建和浏览，并且这个虚拟场景还可以与 MATLAB 交互，因而是可控的。对于广大专业工程技术研究人员来说，这无疑是一个福音。

在“数字化地球”的时代，电子地图的制作是热点。MATLAB 的地图制作工具箱提供了制作电子地图的一种途径。该工具箱的功能强大，可以绘制三维地图。

在编写过程中，作者力求全书思路清晰，结构合理，叙述流畅，术语地道，实例丰富，并诚挚地希望能收到抛砖引玉的效果。如果你看了书以后很有想法，我们可以交流；如果很有收获，甚至做出一个很好的三维系统，我们愿意分享你的快乐！

本书适合于对 MATLAB 感兴趣的大学生、研究生、教师和科研技术人员阅读。

写作过程中得到了很多读者朋友和网友的热心支持，表示感谢！另外，还要感谢黄国明、刘波、王卫、刘玉珊等给予的帮助！

由于水平有限，书中缺点和错误之处在所难免，谨请读者朋友批评指正！可通过电子邮件与我们联系：

苏金明　s_jm@263.net.cn

王永利　wangyl@cdut.edu.cn

编　著　者

目　录

第 17 章　图像处理工具箱简介

随着数字化时代的来临，图像处理知识显得越来越重要。实际上，图像处理已经渗透到计算机、电子、电信、地质、气象、医学等诸多领域。MATLAB 的图像处理工具箱提供了较多的图像处理功能，而且，由于工具箱采用的数据类型与 MATLAB 的相兼容，在工具箱中也可以利用 MATLAB 强大的数值计算能力，从而为图像处理自定义算法的实现提供了快速实现的可能性。

利用图像处理工具箱，可以完成以下任务：

- 图像合成　可以实现图像的代数运算和逻辑运算；
- 空间变换　可以对图像进行旋转、缩放和裁剪等操作；
- 邻域和块处理　可以进行块处理操作、滑动邻域操作、分离块操作和列处理；
- 线性滤波和滤波器设计　可以进行线性滤波和设计 FIR 等滤波器；
- 基于区域进行处理　可以指定区域并对区域进行滤波和填充；
- 变换域处理　可以进行傅里叶变换、离散余弦变换和 Radon 变换；
- 数学形态学运算　可以进行膨胀和腐蚀，以及基于膨胀和腐蚀处理，可以进行数学形态学重建等操作；
- 图像分析　可以进行灰度统计、边缘检测、边界跟踪和四叉树分解等操作；
- 图像增强　可以进行灰度调整和去噪处理；
- 图像配准　可以基于控制点配准图像；
- 图像恢复　可以利用各种滤波器和算法恢复图像。

17.1　图像类型

图像处理工具箱支持索引图像、灰度图像、二值图像和 RGB 图像等 4 种基本的图像类型。下面讨论 MATLAB 和图像处理工具箱如何表示这几种图像。

17.1.1　索引图像

索引图像由数据矩阵 ***X*** 和映射矩阵 ***map*** 组成。数据矩阵可以是 uint8, uint16 或 double 型的。映射矩阵是一个 double 型的 $m\times 3$ 数组，元素为[0,1]范围内的浮点值。***map*** 矩阵的每一行指定某种颜色的红色、绿色和蓝色组分。索引图像将像素值与 ***map*** 矩阵的值直接进行映射。每个图像元素的颜色是通过将 ***X*** 的对应值作为索引编号，从 ***map*** 矩阵中得到的。值 1 指向 ***map*** 矩阵的第一行，值 2 指向第二行，依此类推。

映射矩阵通常与索引图像一起保存，并且在用 imread 函数载入图像时自动载入。但是，并不局限于使用默认的映射方式——可以使用你选择的任何映射方式。下面的图形演示了索引图像的结构。图像中的像素用整型值表示，它们作为索引编号用于获取保存在颜色查

找表中的颜色值。图 17-1 描述一幅索引图像。

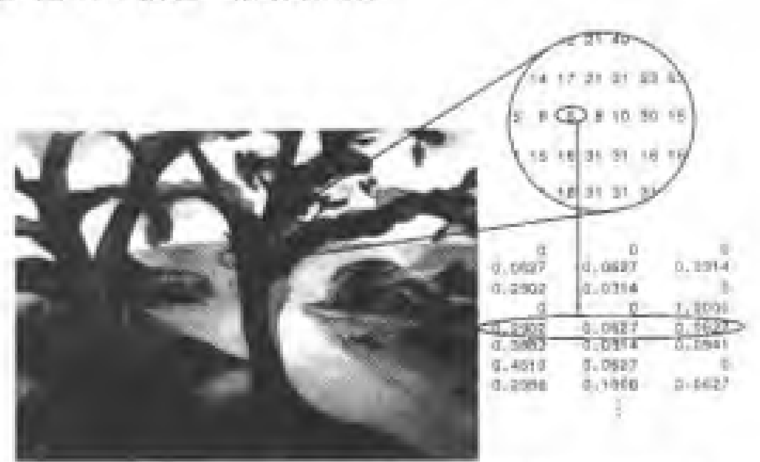


图 17-1　索引图像中像素值与颜色查找表之间的关系

图像矩阵和颜色查找表中值之间的关系取决于图像矩阵的类型。如果图像矩阵是 double 型的，值 1 指向颜色查找表的第一行，值 2 指向第二行，依此类推。如果图像矩阵的 uint8 或 uint16 类型的，就会存在偏移——值 0 指向颜色查找表的第一行，值 1 指向第二行，依此类推。

偏移还用在图形文件格式中，使得可以支持的颜色个数最大化。在前面的图像中，图像矩阵是 double 类型的。因为没有偏移，值 5 指向颜色查找表的第 5 行。

注意，对于 uint16 类型，工具箱只提供了有限的支持。可以将这种类型的数据读入 MATLAB 并且显示它们，但是在处理 uint16 类型的索引图像以前，必须首先将它转换为 double 型或 uint8 型。要转换为 double 型，调用 im2double 函数；要将图像转换为 256 色，调用 imapprox 函数。

17.1.2　灰度图像

灰度图像是一个数据矩阵 *I*，其值表示一定范围内的亮度值。MATLAB 将一幅灰度图像保存为一个单一的矩阵，矩阵的每个元素对应于一个图像像素。矩阵可以是 double, uint8 或 uint16 型的。

亮度矩阵中的元素表示不同的亮度或灰度级，其中亮度 0 通常表示黑色，亮度 1 255 或 65 535 通常表示饱和亮度或白色。

图 17-2 描述一个 double 型灰度图像。

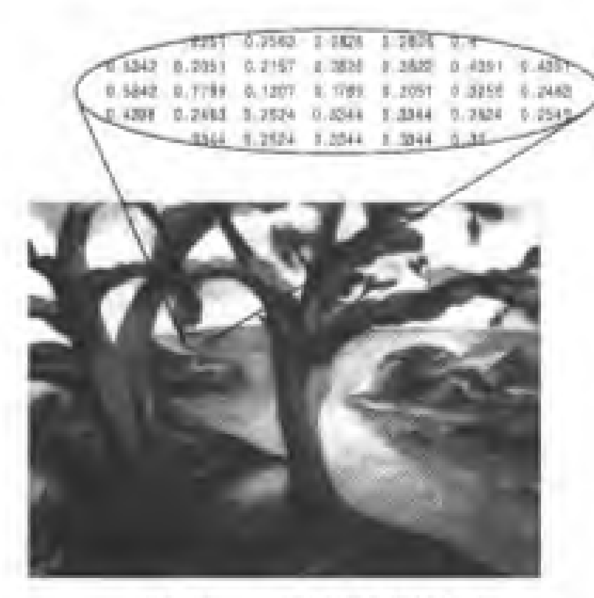


图 17-2　灰度图像中的像素值

17.1.3　二值图像

在二值图像中，假设每个像素取两个离散值中的一个。这两个值对应于 on 和 off。二值图像保存为 logical 数组，值为 0 和 1。

图 17-3 描述一幅二值图像。

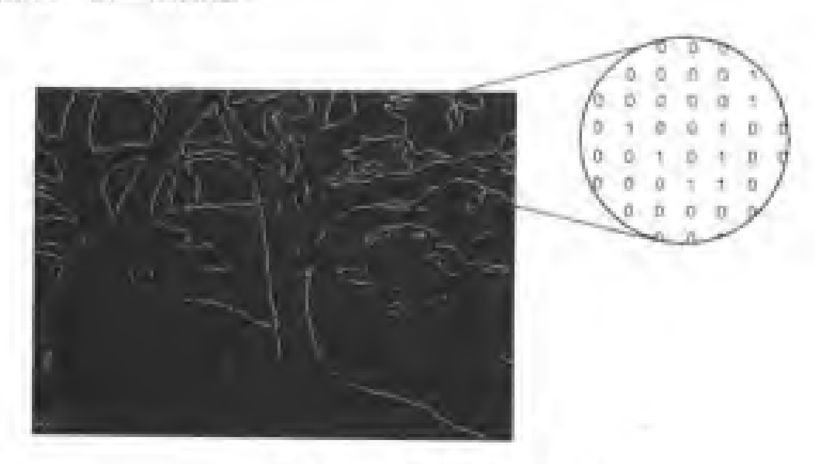


图 17-3　二值图像中的像素值

17.1.4　RGB 图像

RGB 图像有时称为真彩色图像，在 MATLAB 中保存为 $m\times n\times 3$ 的数据数组，定义每个单独像素的红色，绿色和蓝色组分。RGB 图像不使用调色板。每个像素的颜色由像素位置上红色、绿色和蓝色亮度的组合确定。RGB 图像是 24 位图像，其中红色、绿色和蓝色组分均为 8 位。这将产生一千六百多万种颜色。采用这些颜色，在精度上可以逼近现实场景中图像的真实颜色。所以，RGB 图像又称为真彩色图像。

RGB 数组可以是 double,uint8 或 uint16 型的。在 double 类型的 RGB 数组中，每一个颜色组分的值取 0 和 1 之间的数。一个颜色组分为（0, 0, 0）的像素显示为黑色，颜色组分为（1, 1, 1）的像素显示为白色。每个像素的这 3 种颜色组分保存在数据数组的第三维上。例如，像素（10, 5）的红色、绿色和蓝色组分分别保存在 RGB（10, 5, 1），RGB（10, 5, 2）和 RGB（10, 5, 3）。

图 17-4 描述一幅 double 型的 RGB 图像。



图 17-4　RGB 图像的颜色面板

要知道像素（2,3）处的颜色，可以查看保存在（2,3,1：3）中的 3 个一组的 RGB 值。假设（2,3,1）包含值 0.5176，（2,3,2）包含值 0.1608，（2,3,3）包含值 0.0627，则像素（2,3）处的颜色为

0.5176 0.1608 0.0627

为了进一步演示 RGB 图像中用到的 3 个单独颜色面的概念，下面的代码示例创建一个简单的包含红色、绿色和蓝色的连续区域的 RGB 图像，然后为它的每个颜色面板创建一幅图像。它单独显示每一幅颜色面板图像和原始图像。

```
RGB=reshape(ones(64,1)*reshape(jet(64),1,192),[64,64,3]);
R=RGB(:,:,1);
G=RGB(:,:,2);
B=RGB(:,:,3);
imshow(R)
figure, imshow(G)
figure, imshow(B)
figure, imshow(RGB)
```

结果如图 17-5 所示。

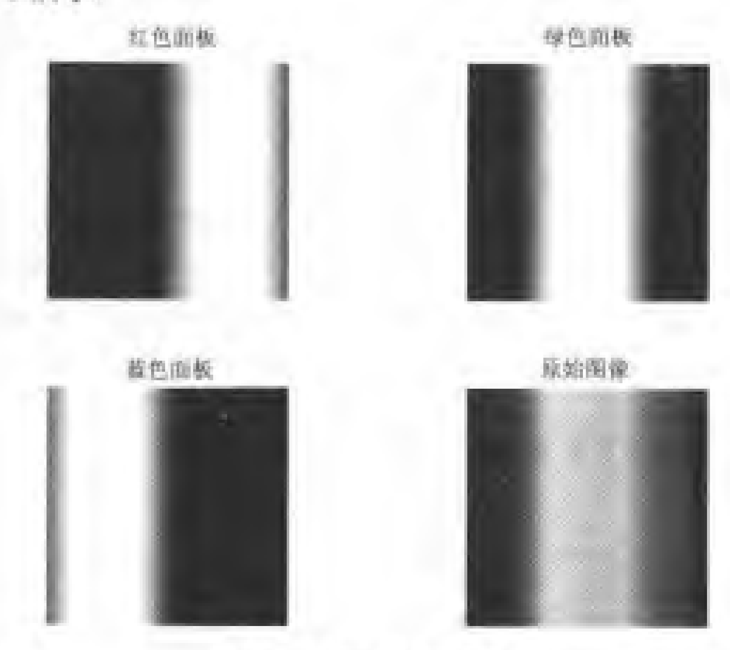


图 17-5　RGB 图像的颜色面板

注意，图像中每个单独的颜色面板中包含一块白色的区域。白色对应于每种颜色的最高值。如，在红色面板图像中，白色表示纯红色值最高。当红色与绿色或蓝色混合时，呈现灰色。图像中的黑色区域显示像素值中不包含红色组分，即 R=0。

17.1.5　图像类型转换

对于有些操作，需要将图像转换为不同的图像类型。例如，如果试图对一幅保存为索引格式的彩色图像进行滤波，应该首先将它转换为 RGB 格式。对 RGB 图像使用滤波器时，MATLAB 将对图像中的亮度值进行滤波，直到合适为止。如果试图对索引图像滤波，

MATLAB 会简单地对索引图像矩阵中的索引值进行滤波，结果可能没有意义。

注意，如果将图像从一种格式转换为另一种格式，产生的图像看起来可能会与原来的图像不同。例如，如果把一幅彩色索引图像转换为灰度图像，产生的图像将可能是彩色图像，而不是灰度图像。

表 17-1 列出了图像处理工具箱中所有的图像转换函数。

表 17-1　图像转换函数

函　数	描　述
dither	采用抖动的方法，把灰度图像转换为二值图像，或者把 RGB 图像转换为索引图像
gray2ind	把灰度图像转换为索引图像
grayslice	利用给定的灰度阈值，将灰度图像转换为索引图像
im2bw	利用给定的灰度阈值，将灰度图像、索引图像或 RGB 图像转换为二值图像
ind2gray	将索引图像转换为灰度图像
ind2rgb	将索引图像转换为 RGB 图像
mat2gray	通过比例化数据，将矩阵中的数据转换为灰度图像
rgb2gray	将 RGB 图像转换为灰度图像
rgb2ind	将 RGB 图像转换为索引图像

还可以用 MATLAB 语句进行一定的图像转换操作。例如，可以通过沿第三维聚合原始矩阵的 3 个拷贝把一幅灰度图像转换为 RGB 格式。

```
RGB = cat(3,I,I,I);
```

红色、绿色和蓝色面板具有相同的矩阵，所以该 RGB 图像显示为灰色。

除了这些标准的转换工具，还有一些函数返回不同类型的图像。

17.2　图像数据

包括图像数据保存类型和图像数据读写等方面的内容。

17.2.1　图像的数据保存类型

MATLAB 中基本的数据结构是数组，它表示一个经过排序的实数或虚数元素集合。自然，它适用于图像数据的表示，因为图像数据是一系列经过排序的颜色或灰度数据的集合。

MATLAB 将大部分图像保存为二维数组（即矩阵），其中，数组的一个元素对应于所显示的图像中的一个像素。例如，一个由 200 行 300 列不同颜色的点组成的图像，在 MATLAB 中保存为一个 200×300 的矩阵。有些图像，比如 RGB 图像，需要用一个三维数组进行保存，其中第三维的第 1 个面表示红色的像素亮度，第 2 个面表示绿色的像素亮度，第 3 个面表示蓝色的像素亮度。

这一特点使得在 MATLAB 中处理图像与处理其他矩阵数据类型类似，并且使得 MATLAB 的所有能力都可用于图像处理。例如，可以用一般的矩阵脚标索引从图像矩阵中选择一个单独的像素。比如，下面的命令返回图像 I 中第 2 行第 15 列处像素的值。

```
I(2,15)
```

默认时，MATLAB 把大部分数据保存为 double 型数组。这些数组中的数据保存为双精度浮点型。所有 MATLAB 函数都可以处理这些数组。

但是，对于图像处理，这一数据表示方式并不总是理想的。图像中的像素个数可以很大，如，一个 1000×1000 的图像有一百万个像素。因为每个像素至少表示一个数组元素，所以这幅图像将需要大约 8MB 内存空间。

为了减小内存需求，MATLAB 支持将图像数据保存为 8 位或 16 位无符号整型数组。这些数组只需要 double 型数组八分之一或四分之一的内存。

对于 uint8 和 uint16 两种类型的图像数据，可以进行许多标准的 MATLAB 数组操作，包括：

- 索引，包括逻辑索引；
- 重塑、排序和聚合；
- 从 MAT 文件读取数据或者把数据写入 MAT 文件；
- 使用关系操作符。

有些 MATLAB 函数，如 find,all,any,conv2,convn,fft2,fftn 和 sum 函数接收 uint8 或 uint16 类型的数据，但是返回值为双精度格式。

但是，基本的算术操作符不接收 uint8 或 uint16 类型的数据。例如，试图对两个 uint8 类型的图像 A 和 B 进行相加操作，将产生下面的错误。

```
C = A + B
??? Function '+' not defined for variables of class 'uint8'.
```

因为这些算术操作是许多图像处理操作的一个重要部分，所以图像处理工具箱中包含了支持对 uint8,uint16 及其他类型数据进行这些操作的函数。

表 17-2 概括了 MATLAB 根据不同的图像类型和存储类型将数据矩阵元素解释为像素颜色的方式。

表 17-2　矩阵元素解释为像素颜色的方式

图像类型	保存类型	解　释
二值图像	logical	元素值为 0 和 1 的数组
索引图像	double	元素值为[1, p]中整数的数组
	uint8 或 uint16	元素值为[0, p-1]中整数的数组
灰度图像	double	元素值为浮点值的数组。值一般在[0, 1]中取值
	uint8 或 uint16	整型数组。值一般在[0, 255]或[0, 65535]中取值
RGB 图像	double	$m\times n\times 3$ 的数组，元素值为[0, 1]中的浮点值
	uint8 或 uint16	$m\times n\times 3$ 的数组，元素为[0, 255]或[0, 65535]中的整型值

17.2.2　读写图像数据

下面介绍如何读写图像数据。内容包括：

- 读取多种不同标准图形文件格式保存的数据；
- 将数据写成多种不同标准图形文件格式；
- 查询图形图像文件，寻找保存在头字段中的信息；
- 转换图像的存储类型；

- 转换图像的文件格式。

1. 读取图像数据

imread 函数从任何受支持的图形图像文件格式和位深读取图像，大部分图像文件格式使用 8 位来保存像素值。读入内存时，MATLAB 把它们保存为 uint8 型。对于支持 16 位数据的文件格式，如 PNG 和 TIFF，MATLAB 把图像保存为 uint16 类型。

注意，对于索引图像，imread 函数总是把颜色映射读入到一个 double 型数组，即使图像数组自身是 uint8 或 uint16 类型时也是如此。

例如，下面的代码将一幅 RGB 图像读入到 MATLAB 工作空间，并用变量 RGB 表示。

```
RGB = imread('football.jpg');
```

本例中，imread 函数从文件的内容推测文件所使用的格式。也可以将文件格式作为一个变量指定给 imread 函数。MATLAB 支持许多通用的图形文件格式，如 BMP，GIF，JPEG，PNG 和 TIFF 等。

MATLAB 支持几种图形文件格式，如 HDF 和 TIFF，它们可以包含多幅图像。默认时，imread 函数只从文件中输入第一幅图像。要从文件中输入其他图像，需要使用文件格式支持的语句。

例如，使用 TIFF 文件时，可以将一个索引值与 imread 函数一起使用来确定文件中希望输入的图像。本例从一个 TIFF 文件中读取 27 幅系列图像并把图像保存到一个四维数组中。可以用 imfinfo 函数确定文件中保存的图像幅数。

```
mri = uint8(zeros(128,128,1,27));    % 预分配四维数组

for frame=1:27
    [mri(:,:,:,frame),map] = imread('mri.tif',frame);
end
```

当文件含有多幅相关图像，比如时间序列图像时，可以将它们保存为四维数组。要求所有图像的大小相同。

2. 写图形图像数据

函数 imwrite 将图像按某种支持的格式写入图形文件。imwrite 函数的大部分语法要求把图像名和文件名作为变量。如果文件名中包含扩展名，则 MATLAB 根据该扩展名推测需要的文件格式。

本例从一个 MAT 文件载入索引图像 X 和它的相关颜色查找表 map，然后把图像作为一幅位图写到文件中。

```
load clown
whos
  Name        Size        Bytes   Class
  X           200x320     512000  double array
  caption     2x1         4       char array
  map         81x3        1944    double array

Grand total is 64245 elements using 513948 bytes
imwrite(X,map,'clown.bmp')
```

与某些图形格式一起使用 imwrite 函数时，可以指定其他参数。例如，对于 PNG 文件，可以把位深作为其他参数。下面的例子把一幅灰度图像 I 写到一个 4 位 PNG 文件中。

```
imwrite(I,'clown.png','BitDepth',4);
```

下面的例子将图像 A 写入到一个 JPEG 文件中，用其他参数来指定压缩质量参数。

```
imwrite(A, 'myfile.jpg', 'Quality', 100);
```

在某些文件格式中，二值图像可以保存为 1 位格式。如果文件格式支持它，默认时 MATLAB 会将二值图像写成 1 位图像。按 1 位图像读入二值图像时，MATLAB 在工作空间中用一个 logical 数组表示它。

下面的例子读入一幅二值图像并将它写到一个 TIFF 文件中。因为 TIFF 格式支持 1 位图像，文件按 1 位格式写入磁盘。

```
BW = imread('text.png');
imwrite(BW,'test.tif');
```

要修改 test.tif 的位深，调用 imfinfo 函数并核对 BitDepth 字段。

```
info = imfinfo('test.tif');
info.BitDepth
ans =
     1
```

注意，写二值文件时，MATLAB 将 ColorType 字段设置为 grayscale。

imwrite 函数使用表 17-3 所示的规则来确定输出图像的存储类型。

表 17-3　确定输出图像存储类型的规则

图像的存储类型	输出图像文件的存储类型
logical	如果指定的输出图像文件格式支持 1 位图像，imwrite 函数创建一个 1 位图像文件。如果指定的输出图像文件格式不支持 1 位图像，如 FPEG，imwrite 函数将图像转换为 uint8 灰度图像类型
uint8	如果指定的输出图像文件格式支持 8 位图像，imwrite 函数创建一个 8 位图像文件
uint16	如果指定的输出图像文件格式支持 16 位图像（PNG 或 TIFF），imwrite 函数创建一个 16 位的图像文件。如果指定的输出图像文件格式不支持 16 位图像，imwrite 函数将图像数据转换为 uint8 类型并创建一个 8 位图像文件
double	ATLAB 将图像数据转换为 uint8 型并创建一个 8 位图像文件，因为大部分图像文件格式使用 8 位

3. 查询一个图形文件

使用函数 imfinfo 可以获得图形文件的信息。获取的信息与文件类型有关，但至少包括下面的内容：

- 文件名；
- 文件格式；
- 文件格式的版本号；
- 文件修改日期；
- 文件大小，按字节计；
- 图像宽度，按像素计；
- 图像高度，按像素计；
- 每像素的位数；
- 图像类型：RGB 图像（真彩色）、灰度图像（灰度）或索引图像。

4．转换图像的存储类型

使用 MATLAB 函数 double，可以将 uint8 和 uint16 型数据转换为 double 双精度型。但是，存储类型之间的转换改变了 MATLAB 和工具箱解释图像数据的方式。如果希望生成的数组被合理解释为图像数据，则需要在进行转换时调整和平衡数据。

为了便于转换存储类型，使用下面工具箱函数中的一个：im2double, im2uint8 或 im2uint16。这些函数自动控制原始数据的调整和平衡。如，下面的命令将一幅数据值在[0, 1]范围内的双精度 RGB 图像转换为一幅数据大小在[0,255]范围内的 uint8 型 RGB 图像。

```
RGB2 = im2uint8(RGB1);
```

把图像转换为位数更低的类型时，通常会丢失一些图像信息。例如，一个 uint16 型灰度图像可以存储最多 65 536 种不同的灰度，但 uint8 型灰度图像只能存储 256 种不同的灰度。把 uint16 型灰度图像转换为 uint8 型灰度图像时，im2uint8 函数会量子化原始图像中的灰度色。换句话说，原始图像中所有从 0 到 127 的值在 uint8 型图像中都会变成 0，从 128 到 385 的值变成 1，依此类推。通常情况下，这种信息的丢失不会造成太大的问题，因为 256 仍然超出了人的肉眼可以识别的灰度级别数。

将一幅索引图像从一种存储类型转换为另一种类型并不总是可行的。在索引图像中，图像矩阵只包括索引值而不是颜色数据本身，所以在转换过程中无法进行颜色数据的量子化。

例如，一幅具有 300 种颜色的 uint16 或 double 型索引图像无法转换为 uint8 型，因为 uint8 型数组只有 256 种不同的值。如果想进行转换，必须首先用 imapprox 函数减少图像中的颜色个数。本函数对颜色查找表中的颜色进行量子化，以减少图像中可以分辨的颜色种数。

5．转换图像的文件格式

如果想改变图像的文件格式，用 imread 函数读入图像，然后用 imwrite 函数指定合适的格式，保存图像。

下面进行演示，用 imread 函数读入一幅 BMP 图像到工作空间中，然后将这个 BMP 图像用 PNG 格式保存到一个文件中。

```
bitmap = imread('mybitmap.bmp','bmp');
imwrite(bitmap,'mybitmap.png','png');
```

17.2.3　读写 DICOM 文件

图像处理工具箱支持对 DICOM 格式图像数据的操作。下面介绍如何完成以下操作：

- 从 DICOM 文件中读取图像数据；
- 从 DICOM 文件中读取元数据；
- 将图像数据写入 DICOM 文件；
- 将元数据写入 DICOM 文件。

1．从 DICOM 文件中读取图像数据

用 dicomread 函数从 DICOM 文件中读取图像数据。dicomread 函数可以读取遵守 DICOM 规范的文件，但也可以读取不遵守 DICOM 规范的文件。

下面的例子从一个 DICOM 文件中读取图像。

```
I = dicomread('CT-MONO2-16-ankle.dcm');
```

使用工具箱中的图像显示函数 imshow 或 imshow 察看图像数据。

```
imview(I,[])
```

生成图 17-6。



图 17-6 显示图像

2．从 DICOM 文件中读取元数据

DICOM 文件包括称为元数据的信息，它描述了图像的特点，比如大小、维数和位深等。另外，DICOM 规范定义了很多其他元数据字段来描述数据的许多其他特点，如用于创建数据的形态特征、用于捕获图像的装备设置和与学习有关的信息等。dicomread 函数可以处理几乎所有由 DICOM 规范定义的元数据字段。

使用 dicominfo 函数从 DICOM 文件中读取元数据。该函数将元数据作为一个结构返回，其中结构中的每个字段是 DICOM 元数据中的一个详细条目。

```
info = dicominfo('CT-MONO2-16-ankle.dcm');
info =
              Filename: [1x47 char]
           FileModDate: '24-Dec-2000 19:54:47'
              FileSize: 525436
                Format: 'DICOM'
         FormatVersion: 3
                 Width: 512
                Height: 512
              BitDepth: 16
             ColorType: 'grayscale'
        SelectedFrames: []
            FileStruct: [1x1 struct]
```

```
             StartOfPixelData: 1140
        MetaElementGroupLength: 192
   FileMetaInformationVersion: [2x1 double]
      MediaStorageSOPClassUID: '1.2.840.10008.5.1.4.1.1.7'
   MediaStorageSOPInstanceUID: [1x50 char]
            TransferSyntaxUID: '1.2.840.10008.1.2'
       ImplementationClassUID: '1.2.840.113619.6.5'
                        ⋮
```

可以使用 dicominfo 函数返回的元数据结构指定 DICOM 文件。例如，可以用下面的代码从示例 DICOM 文件中读取元数据，然后将元数据传递给 dicomread 函数，从文件中读取图像数据。

```
info = dicominfo('CT-MONO2-16-ankle.dcm');
I = dicomread(info);
```

3．写数据到 DICOM 文件

按 DICOM 格式写图像数据到文件，需要使用 dicomwrite 函数。下面的例子将图像 I 写到 DICOM 文件 ankle.dcm 中。

```
dicomwrite(I,'h:\matlab\tmp\ankle.dcm')
```

4．写元数据到 DICOM 文件

写图像数据到 DICOM 文件时，dicomwrite 函数包括了元数据字段的最小设置，这个最小设置是创建 DICOM 信息对象所必需的。dicomwrite 函数支持 3 种类型的 DICOM 信息对象，即

- 二次俘获（默认）;
- 核磁共振;
- X 线断层摄影术。

将一个已经存在的 DICOM 元数据结构传递给 dicomwrite 函数，可以把希望写入的元数据指定给要保存的文件。DICOM 元数据结构可以用 dicominfo 函数提取。

```
info = dicominfo('CT-MONO2-16-ankle.dcm');
I = dicomread(info);
dicomwrite(I,'h:\matlab\tmp\ankle.dcm',info)
```

此时，dicomwrite 函数将元数据结构中的信息写入到新的 DICOM 文件中。将元数据写入文件时，有一些字段是 dicomwrite 函数必须更新的。例如，dicomwrite 函数必须更新新文件中的文件修改日期。作为演示，下面对原始元数据中的文件修改日期和新文件中的文件修改日期进行比较。

```
info.FileModDate
ans =
24-Dec-2000 19:54:47
```

使用 dicominfo 函数，从新写的文件中读取元数据并且核对文件修改日期。

```
info2 = dicominfo('h:\matlab\tmp\ankle.dcm');
info2.FileModDate
ans =
16-Mar-2003 15:32:43
```

第 18 章 显示图像

MATLAB 中包括两个图像显示函数：image 和 imagesc。这两个函数都创建句柄图形对象中的 Image 对象并且有设置不同属性的语法格式。imagesc 函数会自动对输入数据进行比例化。

图像处理工具箱包括两个显示函数：imview 和 imshow。通常，使用这两个函数比使用 image 和 imagesc 函数效果更好，因为它们更容易使用并且经过优化处理更适合显示图像。

18.1 用图像查看器显示图像

利用图像查看器可以显示图像。图像查看器在单独的窗口中显示图像并提供图像的大小信息、像素值的显示范围和鼠标光标处的像素值。另外，图像查看器还提供了 3 个其他工具：

- 概览窗口　该窗口将整幅图像显示在一个小的单独窗口中。在概览窗口中，显示在图像浏览器中的图像部分用一个称为细节矩形的红色矩形显示。移动该矩形，可以改变图像查看器窗口中的显示内容。
- 像素区域工具　该工具检查图像指定区域中的像素值。在图像上拖拉像素区域矩形，可以选择区域。像素区域工具在一个单独的窗口中将像素值显示在该区域内。这个工具使得在图像中指定可视元素的操作更容易了。
- 图像信息窗口　显示像素值等图像信息。

图 18-1 显示了图像查看器和它的工具。

下面介绍如何在图像查看器中打开一幅图像。主要包括：

- 启动图像查看器；
- 查看多幅图像；
- 指定图像初始大小；
- 关闭图像查看器。

1．启动图像查看器

调用 imview 函数，指定要查看的图像，启动图像查看器。可以用 imview 函数显示已经导入到 MATLAB 工作空间的图像。

```
moonfig = imread('moon.tif');
imview(moonfig);
```

还可以指定包含图像的文件的名称，如下所示：

```
imview('moon.tif');
```

该文件必须位于当前目录或 MATLAB 路径中。这个语法格式对于扫描多幅图像比较有用，但是图像数据不会保存到 MATLAB 工作空间中。

如果调用 imview 函数时没有指定任何变量，则显示一个文件选择对话框。

图18-1 图像查看器和相关工具

2. 查看多幅图像

如果指定一个包含多幅图像的文件，imview 函数只显示文件中的第 1 幅图像。要查看文件中的所有图像，需要用 imread 函数将每幅图像导入到 MATLAB 工作空间，然后多次调用 imview 函数，每次显示一幅图像。

有些应用程序创建与时间或视图有关的图像集合，如核磁共振成像的切片图或取自视频流数据的帧。图像处理工具箱将这些图像集合处理成四维数组，其中每一个单独的图像称为一帧，所有这些帧在第四维上聚合。多帧图像上的所有帧必须是大小相同的。图像查看器一次只显示一幅图像。使用标准的 MATLAB 索引语法指定要显示的帧。

```
imview(multiframe_array(:,:,:,1));
```

要想一次就能查看到多帧图像中的所有帧，需要使用 montage 函数。

3. 指定初始图像大小

默认时，imview 函数显示图像时的放大倍率为 100%。这里，100%表示 imview 函数将图像上的每个像素映射到一个屏幕像素上。通常这是显示图像的最好方法。但是，在某些情况下，特别是操作小图像时，可能希望 imview 函数将图像比例化至图像查看器的最小尺寸。

要控制 imview 函数显示的图像的初始放大倍率，使用下面几种方法中的一种：

● 将当前 MATLAB 的 ImviewInitialMagnification 选项设置为“fit”。默认值为 100，指定放大倍率为 100%。

● 将 imview 函数的选项参数“InitialMagnification”设置为“fit”。

```
imshow(X, map,'InitialMagnification','fit')
```

4. 关闭图像查看器

用窗口标题条中的“Close”按钮关闭图像查看器窗口。如果有多个图像查看器窗口是打开的，可以用下面的语法关闭它们。

```
imview close all
```

还可以用 imview 函数返回一个图像查看器的句柄，并使用该句柄关闭图像查看器。

18.2 用 imshow 函数显示图像

本节介绍如何用 imshow 函数显示图像，内容包括：

- 打开图像；
- 指定图像的初始大小；
- 查看多幅图像；
- 理解句柄图形对象属性设置。

18.2.1 打开图像

可以用 imshow 函数查看图像。如下例所示，用 imshow 函数显示已经导入到 MATLAB 工作空间的图像。

```
moon = imread('moon.tif');
imshow(moon);
```

还可以简单地指定包含图像的文件的名称，并将它作为变量传递给 imshow 函数，如下面代码所示。注意，文件必须位于当前路径或 MATLAB 路径中。

```
imshow('moon.tif');
```

该语法对于扫描所有图像比较有用。但是，要注意的是，使用该语法时，图像数据不是保存在 MATLAB 工作空间中的。如果想把图像引入工作空间，必须使用 getimage 函数，它从当前 Image 对象中提取图像。例如，

```
moon = getimage;
```

如果显示 moon.tif 图像的图形窗口当前是激活的，则将图像数据赋给变量 moon。

18.2.2 指定图像的初始大小

在大部分情况下，当工具箱在默认情况下运行时，imshow 函数将一个单独的屏幕像素指定给每个图像像素，例如，一幅 200×300 的图像显示在屏幕上时高为 200 个屏幕像素，宽为 300 个屏幕像素。通常，这是显示图像的更好方法。imshow 函数调用 truesize 命令进行这种图像像素至屏幕像素的映射。

在有些情况下，可能不想 imshow 函数自动调用 truesize 命令（如操作小图像时）。此

时，图像按默认大小显示。要想在不调用 truesize 命令的情况下使用 imshow 函数，按照下面的步骤操作：

- 将 ImshowTruesize 参数设置为 manual；
- 将 imshow 函数的 display_option 参数设置为 notruesize。即

```
imshow(X, map,'notruesize')
```

imshow 函数不使用 truesize 命令时，必须通过插值来确定那些不直接与图像矩阵元素对应的屏幕像素的值。

18.2.3 查看多幅图像

如果将一个包含多幅图像的文件指定给 imshow 函数，它将只显示文件中的第 1 幅图像。要查看文件中的所有图像，调用 imread 函数将图像导入到 MATLAB 工作空间中。

有些应用创建与时间或视图有关的图像集合，如核磁共振切片图或源于视频流数据的帧。图像处理工具箱把这些图像集合作为四维数组进行处理，其中每一幅单独的图像称为一个帧，所有的帧在第四维上进行聚合。多帧图像上的所有帧必须是大小相同的。一旦图像存在于 MATLAB 工作空间中，用 imshow 函数可以有两种方法显示它们：

- 每幅图像用一个单独的图形窗口进行显示；
- 在一个单独的图形窗口中显示多帧。

1. 将每幅图像显示在单独的图形窗口中

最简单的方法就是将每幅图像显示在单独的图形窗口中。MATLAB 本身对可以同时显示的图像数目没有任何限制，当然，该数目与计算机的硬件有关系。imshow 函数总是将图像显示在当前图形窗口中，所以，如果连续显示两幅图像，则第 2 幅图像会覆盖第 1 幅图像。要避免这种情况出现，必须在调用 imshow 函数以前用 figure 命令创建一个新的空图形窗口。例如，下面的代码查看表示灰度图像 I 的数组中第 1 个 3 帧图像。

```
imshow(I(:,:,:,1))
figure, imshow(I(:,:,:,2))
figure, imshow(I(:,:,:,3))
```

使用这个方法时，最初图形窗口是空的。

2. 在同一图形窗口中显示多幅图像

可以将 imshow 函数与 MATLAB 的 subplot 函数或 subimage 函数结合使用，在一个单独的图形窗口中显示多幅图像。

subplot 函数将一个图形窗口分割成多个显示区域，该函数的语法为

```
subplot(m,n,p)
```

该语法将图形窗口分割成 $m \times n$ 块显示区域并使第 p 块显示区域激活。例如，如果要并排显示两幅图像，键入类似下面的命令行，

```
[X1,map1]=imread('forest.tif');
[X2,map2]=imread('trees.tif');
subplot(1,2,1), imshow(X1,map2)
subplot(1,2,2), imshow(X2,map2)
```

其结果如图 18-2 所示。



图 18-2　并排显示两幅图像

使用 subplot 函数会共用一个颜色映射矩阵。如果共用一个颜色映射矩阵生成了不可接受的显示结果，则可以使用 subimage 函数，或者在载入图像时将所有图像映射到同一颜色映射矩阵。

subimage 函数在显示图像之前将图像转换为 RGB 格式，从而避免了共用颜色映射矩阵的问题。下例显示与图 18-2 中相同的两幅图像。

```
[X1,map1]=imread('forest.tif');
[X2,map2]=imread('trees.tif');
subplot(1,2,1), subimage(X1,map1)
subplot(1,2,2), subimage(X2,map2)
```

结果如图 18-3 所示。显然，显示效果更佳。

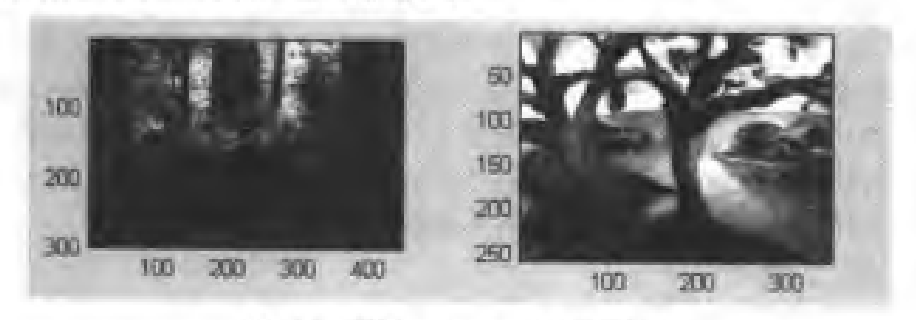


图 18-3　使用 subimage 函数显示图像

18.2.4　理解句柄图形对象的属性设置

显示索引图像、灰度图像、二值图像或 RGB 图像时，imshow 函数通过设置句柄图形的属性来控制图像的显示。表 18-1 列出了每种图像的相关属性和它们的设置。该表用标准工具箱术语表示不同的图像类型：X 表示索引图像，I 表示灰度图像，BW 表示二值图像，而 RGB 表示 RGB 图像。

图 18-1　各种图像的相关属性及其设置

句柄图形属性	索引图像	灰度图像	二值图像	RGB 图像
CData (Image)	设置为 X 中的数据	设置为 I 中的数据	设置为 BW 中的数据	设置为 RGB 中的数据
CDataMapping (Image)	设置为'direct'	设置为 'scaled'	设置为'direct'	CData 为三维时忽略
CLim (Axes)	不采用	double: [0 1] uint8: [0 255] uint16: [0 65535]	设置为 [0 1]	CData 为三维时忽略

续表

句柄图形属性	索引图像	灰度图像	二值图像	RGB图像
Colormap (Figure)	设置为map中的数据	设置为grayscale颜色映射矩阵	设置为grayscale颜色映射矩阵，值的范围为黑到白	CData为三维时忽略

18.3 显示不同类型的图像

下面介绍如何用imshow函数和imview函数显示不同类型的图像。

18.3.1 显示索引图像

用imshow函数或imview函数可以显示索引图像，需要指定图像矩阵和颜色映射矩阵。即

```
imshow(X,map)
```

或

```
imview(X,map)
```

对于X中的每个像素，这些函数都会显示保存在map对应行中的颜色。如果图像矩阵数据是double型的，则值1指向颜色映射矩阵的第1行，值2指向颜色映射矩阵的第2行，依此类推。但是，如果图像矩阵数据是uint8型或uint16型的，则值0指向颜色映射矩阵的第1行，值1指向第2行，依此类推。这种偏离imview函数和imshow函数会自动控制。

如果颜色映射矩阵包含的颜色数目比图像中的多，则函数删除多余的颜色。如果颜色映射矩阵中的颜色数目比图像中的少，则函数将所有超出的图像像素设置为颜色映射矩阵中的最后一种颜色。例如，如果一幅uint8型的图像包含256种颜色，现在用一个只有16种颜色的颜色映射矩阵去显示它，则所有值大于或等于15的像素都用颜色映射矩阵中的最后一种颜色显示。

18.3.2 显示灰度图像

用imshow或imview函数显示灰度图像，将图像矩阵作为变量。即

```
imshow(I)
```

或

```
imview(I)
```

两个函数都通过将图像的灰度值比例化后作为灰度颜色映射矩阵的索引值来显示图像。如果I是double型的，值为0的像素显示为黑色，值为1的像素显示为白色，二者之间的像素显示为灰色。如果I是uint8型的，则值为255的像素显示为白色。如果I为uint16型的，则值为65 535的像素显示为白色。

灰度图像与索引图像的相似之处在于，它们都使用了一个$m\times3$的RGB颜色映射矩阵。但是，一般情况下都不会给灰度图像指定一个颜色映射矩阵。MATLAB使用灰度系统颜色映射矩阵显示灰度图像。默认时，颜色映射矩阵中的灰度级别数在24位的系统上是

256，在其他系统上是 64 或 32。

使用 imshow 函数，可以有选择地指定灰度图像中的灰度级别数。例如，使用下面的语法，将图像显示为具有 32 个灰度级别。

```
imshow(I,32)
```

因为 MATLAB 对灰度图像进行比例化，使之填满整个颜色映射矩阵范围，所以任何大小的颜色映射矩阵都可以使用。使用更大的颜色映射矩阵可以看到更多的细节。

有的数据落在数据类型规定的范围以外，如 uint8 型数据落在[0, 255]外，uint16 型数据落在[0, 65535]外等。要将这些数据显示成灰度图像，可以用下面的语法直接指定显示范围。

```
imshow(I,[low high])
```

或

```
imview(I,[low high])
```

如果用一个空矩阵表示显示范围，则这两个函数会自动对数据进行比例化，将 low 和 high 设置为数组中的最小值和最大值。下面的例子对一幅灰度图像进行滤波，生成位于规定范围以外的数据，然后调用 imview 函数将该数据显示成图像。

```
I = imread('testpat1.png');
J = filter2([1 2;-1 -2],I);
imview(J,[]);
```

结果如图 18-4 所示。注意图中右下角所示的数据范围。

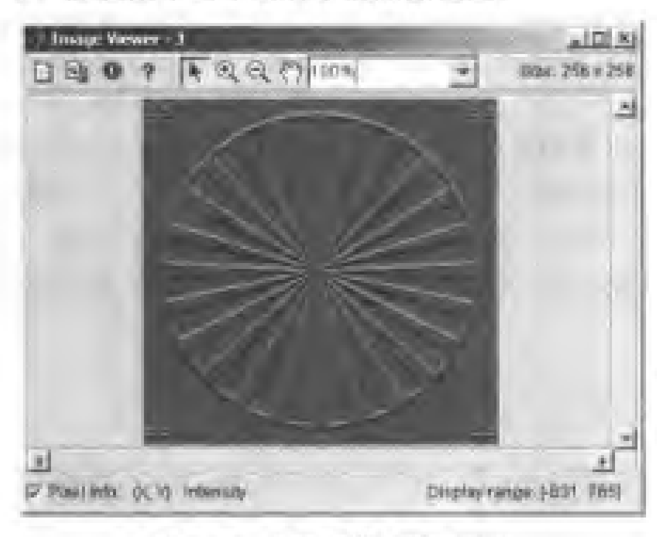


图 18-4　指定图像的灰度范围

18.3.3　显示二值图像

在 MATLAB 中，二值图像是 logical 类型的。用 imshow 函数或 imview 函数显示二值图像，将图像矩阵作为变量。例如，下面的代码将一幅二值图像读入到 MATLAB 工作空间然后显示它。

```
BW = imread('circles.png');
imshow(BW)
```

或

```
imview(BW)
```

结果如图 18-5 所示。

用 MATLAB 中的 Not(~)操作改变颜色。例如，

```
imshow(~BW)
```

或

```
imview(~BW)
```

改变颜色后的图像如图 18-6 所示。

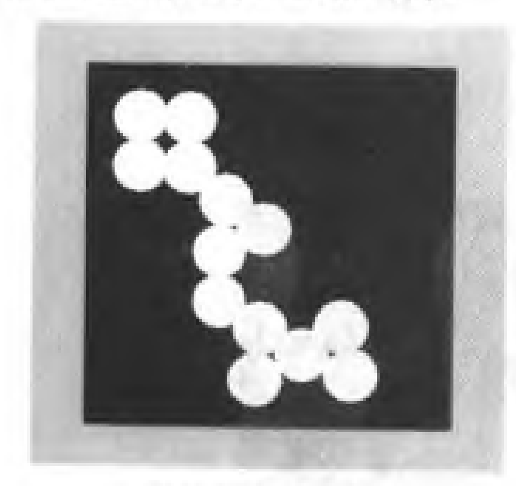


图 18-5 显示二值图像

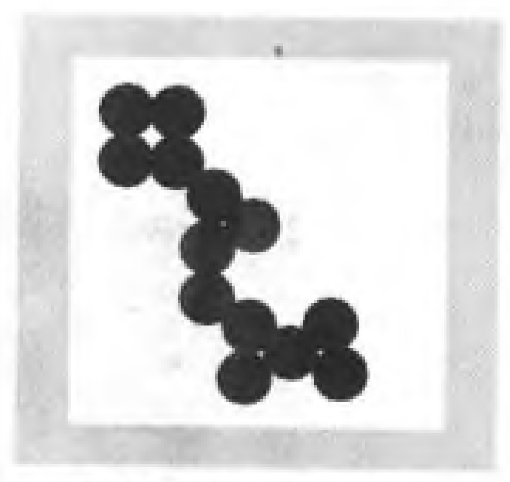


图 18-6 改变二值图像的颜色

还可以用索引图像的颜色映射语法显示二值图像。例如，下面的命令指定一个两行的颜色映射矩阵，矩阵中 0 表示红色，1 表示蓝色。即

```
imshow(BW,[1 0 0; 0 0 1])
```

或

```
imview(BW,[1 0 0; 0 0 1])
```

效果如图 18-7 所示。



图 18-7 用颜色映射显示二值图像

18.3.4　显示 RGB 图像

RGB 图像是一个 $m \times n \times 3$ 的数组。对于图像中的每一个像素（r, c），颜色用 3 个一组的（r, c, 1：3）表示。用 imshow 函数或 imview 函数显示 RGB 图像，将图像矩阵指定为变量。例如，下面的代码将一幅 RGB 图像读入 MATLAB 工作空间，然后显示该图像。

```
RGB = imread('peppers.png');
imshow(RGB)
```

或

```
imview(RGB)
```

显示结果如图 18-8 所示。



图 18-8　一幅 RGB 图像

在 24 位的系统上，可以直接显示真彩色图像，因为它们可以分别给红色、绿色和蓝色层分配 8 位（256 个水平）。在颜色更少的系统上，imshow 函数采用颜色近似和抖动的组合来显示图像。

18.4　特殊显示技巧

除了 imshow 函数和 imview 函数以外，工具箱还提供了进行特殊显示操作或对显示格式进行更直接控制的函数。这些函数与 MATLAB 图形函数一起，提供了多种图像显示方式。

主要内容包括：

- 添加颜色条；
- 一次显示多帧图像的所有帧；
- 将多帧图像转换为动画；
- 纹理映射。

18.4.1 添加颜色条

颜色条用来表示图像灰度值的范围。给图像添加颜色条，首先用 imshow 函数将图像显示在 MATLAB 图形窗口，然后调用 colorbar 函数显示颜色条。给包含图像对象的坐标轴对象添加颜色条时，颜色条表示与图像中不同颜色对应的数据值。不能给图像查看器中显示的图像添加颜色条。

如果要显示的数据在数据类型规定的范围之外，用颜色条查看数据值与颜色之间的对应关系很有用。下面的例子中，对一幅 uint8 型灰度图像进行过滤以后，数据不再落在[0, 255]范围内。

```
RGB = imread('saturn.png');
I = rgb2gray(RGB);
h = [1 2 1; 0 0 0; -1 -2 -1];
I2 = filter2(h,I);
imshow(I2,[]), colorbar
```

结果如图 18-9 所示。



图 18-9 添加颜色条

18.4.2 一次显示多帧图像的所有帧

一次显示多帧图像的所有帧，需要使用 montage 函数。该函数将图形窗口分割成多个显示区域，并将每幅图像显示在一个单独的区域中。

montage 函数的语法与 imshow 函数相近。显示多帧灰度图像的语法为

```
montage(I)
```

显示多帧索引图像的语法为

```
montage(X,map)
```

注意，多帧索引数组中的所有帧必须使用相同的颜色映射矩阵。

下面的例子装载和显示了多帧索引图像中的所有帧。首先初始化一个含有 27 帧图像的数组，这 27 帧图像取自一幅多帧图像文件。然后进行循环，在每一次迭代时用 imread 函数读入一帧。

```
mri = uint8(zeros(128,128,1,27));
for frame=1:27
    [mri(:,:,:,frame),map] = imread('mri.tif',frame);
end
montage(mri,map);
```

显示结果如图 18-10 所示。

montage 函数在第 1 行的第 1 个位置上显示第 1 帧，在第 1 行的第 2 个位置上显示第 2 帧，依此类推。



图 18-10　将多帧图像的所有帧显示在一个图形窗口中

18.4.3　将多帧图像转换为动画

根据多帧图像数组创建 MATLAB 动画，使用 immovie 函数。下面的例子根据多帧索引图像创建一部动画：

```
mov = immovie(X,map);
```

这里，X 是一个用于创建电影的图像四维数组。

可以用 movie 函数播放该动画，即

```
movie(mov);
```

下面的例子载入多帧图像 mri.tif，然后将它转换为动画。

```
mri = uint8(zeros(128,128,1,27));
for frame=1:27
    [mri(:,:,:,frame),map] = imread('mri.tif',frame);
end

mov = immovie(mri,map);
movie(mov);
```

效果如图 18-11 所示。

注意，immovie 函数在电影创建的过程中会进行显示，所以这部电影实际上被看了两次。第 2 次用 movie 函数播放时要快得多。



图 18-11　将多帧图像转换为动画

18.4.4　纹理映射

用 imshow 函数或 imview 函数查看图像时，MATLAB 会对图像进行二维显示。但是，

也可能将一幅图像映射到参数表面（如圆球）上，或表面图下方。warp 函数通过纹理映射来创建这些显示效果。纹理映射是一种利用插值将图像映射到表面网格的处理方法。

下面的例子将一幅测试模式的图像纹理映射到一个柱面上。

```
[x,y,z] = cylinder;
I = imread('testpat1.png');
warp(x,y,z,I);
```

显示结果如图 18-12 所示。

如果纹理映射的效果不够理想，可以通过调整 Xdir,Ydir 和 Zdir 属性设置来进行修改。

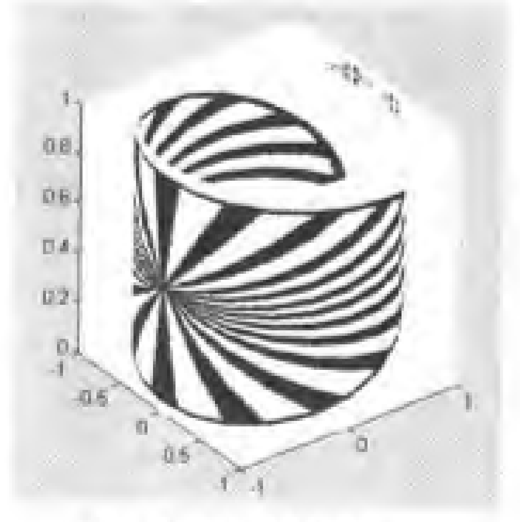


图 18-12 将纹理映射到柱面上

18.5 打印图像

将图像输出到其他应用程序如 Word 等，可以用 imwrite 函数将文件保存为合适的格式。如果要打印图像，用 imshow 函数将图像显示到 MATLAB 图形窗口，然后用 MATLAB 的 print 命令或"File"菜单中的"Print"选项打印图像。从图形窗口中打印时，打印出来的图像包括了图像以外的元素，如标记、标题和其他注释等。

18.6 设置图像显示的参数选项

可以用工具箱的参数选项来控制 imshow 函数和 imview 函数显示图像的效果。例如，可以用 imshow 函数取消图形窗口中坐标轴和标记的显示，或者用 imview 函数指定初始放大倍率。

图像处理工具箱支持几种 imshow 函数和 imview 函数显示图像的参数选项。表 18-2 列出了这些选项并给出了简洁的描述。

图 18-2 工具箱参数选项

工具箱参数选项	描 述
ImshowBorder	控制用 imshow 函数显示图像时，在图像坐标轴与图形窗口边缘之间留出空白还是不留空白
ImshowAxesVisible	控制 imshow 函数显示图像时是否显示坐标轴和标记标签
ImshowTruesize	控制 imshow 函数是否调用 truesize 函数
ImviewInitialMagnification	控制图像查看器第一次显示图像时使用的放大倍率
TrueSizeWarning	控制图像比屏幕还大时是否返回一个警告信息

用 iptgetpref 函数确定参数选项的当前值。下面的例子用 iptgetpref 函数确定 ImviewIntialMagnification 参数选项的值。

```
iptgetpref('ImviewInitialMagnification')
ans =
   100
```

注意，参数选项名是有大小写区分的，并且可以缩写。

用 iptsetpref 函数指定工具箱参数选项的值。下面的例子调用 iptsetpref 函数，指定让 imshow 函数改变图形窗口的大小，使它正好包围显示的图像。

```
iptsetpref('ImshowBorder', 'tight');
```

第 19 章　颜色和坐标

19.1　颜色

本节介绍有助于处理彩色图像数据的工具箱函数，注意，“彩色”包括了灰色，所以本章的许多讨论也适用于灰度图像。

19.1.1　屏幕位深

大部分计算机显示器的屏幕像素采用 8 位、16 位或 24 位。屏幕像素的位数决定显示器的屏幕位深。屏幕位深确定屏幕的颜色分辨率，即显示器可以产生多少种不同的颜色。

不考虑系统可以显示的颜色数目，MATLAB 可以用很高的位深保存和处理图像：uint8 型 RGB 图像采用 2^{24} 色，uint16 型 RGB 图像采用 2^{48} 色，double 型 RGB 图像采用 2^{159} 色。在 24 位颜色的系统上，这些图像的显示效果最佳。但是，通常在 16 位的系统上效果也不错。

要确定系统的屏幕位深，在 MATLAB 提示符后面输入下面的命令行。

```
get(0,'ScreenDepth')
ans =
       32
```

返回的整型值表示每个屏幕像素的位数。

表 19-1 列出了不同位数对应的屏幕位深。

表 19-1　不同位数对应的屏幕位深

值	屏幕位深
8	8 位显示模式支持 256 色。8 位显示模式能生成 24 位显示模式可以生成的任何颜色，但是一次只能显示 256 种不同的颜色
16	16 位显示模式通常对每一种颜色组分使用 5 位，从而使得红色、绿色和蓝色都有 32 个深度级别。它支持 32 768 种颜色。有的系统使用另外 1 位增加绿色的深度级别数，此时，16 位显示模式能支持的颜色数目达到 64 536 种
24	24 位显示模式对每一种颜色组分使用 8 位，使得红色、绿色和蓝色都有 256 个深度级别。它支持 16 777 216 种不同的颜色。用 24 位显示模式可以渲染逼真的图像
32	32 位显示模式用 24 位保存颜色信息，用剩下的 8 位保存透明度数据

根据目前使用的系统来选择屏幕位深。通常，24 位显示模式的使用效果最佳。如果需要使用一个更低的屏幕位深，使用 16 位比使用 8 位更好。但是需要记住的是，16 位有下面一些局限性：

- 图像可能具有比 16 位显示模式表现得更好的颜色级别。如果不能获得某种颜色，MATLAB 用最接近的颜色代替它。
- 只有 32 个灰度级别。如果主要基于灰度图像进行操作，用 8 位显示模式可能能获取

更好的显示结果。

19.1.2 减少图像中的颜色种数

下面介绍如何减少索引图像或 RGB 图像中的颜色种数；同时讨论抖动方面的问题，它在工具箱的减色函数中要用到。抖动技术用于增加图像中当前的颜色数目。

表 19-2 综合列出了图像处理工具箱中的减色函数。

表 19-2 减色函数

函 数	描 述
imapprox	减少索引图像中的颜色数目，使得可以在新的颜色查找表中指定颜色数目
rgb2ind	把 RGB 图像转换为索引图像，使得可以指定颜色数目并保存在新的颜色查找表中

在 24 位显示模式的系统上，RGB（真彩色）图像最多可以显示 16 777 216 种颜色。在低屏幕位深的系统上，RGB 图像的显示效果仍然相当好，因为 MATLAB 会自动在必要的时候使用颜色近似和抖动。

但是，如果颜色数目很大，索引图像可能引起问题。通常，应该将索引图像局限于 256 色，因为：

- 在 8 位显示模式的系统上，256 色以上的索引图像需要被抖动或映射，所以，显示效果不好；
- 在有些平台上，颜色映射不能超过 256 个入口；
- 如果索引图像大于 256 色，MATLAB 不能将图像数据保存到 uint8 数组中，但通常使用一个 double 型数组来代替，造成存储空间的浪费；
- 大部分图像文件格式将索引图像局限于 256 色。如果用 256 种以上的颜色将一幅索引图像写成不支持 256 色以上的格式，将产生错误。

1．使用 rgb2ind 函数

rgb2ind 函数将一幅 RGB 图像转换为索引图像，在处理过程中减少颜色数目。该函数提供了下面一些方法来使得输出图像的颜色尽量与原始图像的接近。

- 量子化，包括均匀量子化和最小方差量子化；
- 颜色查找表映射。

最后生成的图像的质量与选用的近似方法、输入图像的颜色范围和是否使用抖动有关。注意，将不同方法用于不同类型的图像可能效果更好。

（1）量子化

在图像中减少颜色数目涉及到量子化概念。函数 rgb2ind 把量子化作为减色算法的一部分。该函数支持两种量子化方法：均匀量子化和最小方差量子化。

讨论图像量子化时的一个重要术语是 RGB 颜色立方体，它在本节内容中会频繁提到。RGB 颜色立方体是一个为特定数据类型定义的所有颜色的三维数组。因为 MATLAB 中的 RGB 图像可以是 uint8, uint16 或 double 型的，所以存在 3 种可能的颜色立方体定义。例如，如果 RGB 图像是 uint8 型的，每个颜色平面定义了 256 个值，则颜色立方体会定义 2^{24} 种颜色。这个颜色立方体对于所有 uint8 型 RGB 图像都是相同的，不管它们实际上使用了哪些颜色。

uint8, uint16 和 double 型颜色立方体都有相同的颜色范围。换句话说，uint8 型 RGB 图像中最亮的红色看起来与 double 型 RGB 图像中最亮的红色相同。区别在于，double 型 RGB 颜色立方体具有更多的红色深度级别。图 19-1 显示了一幅 uint8 型图像的 RGB 颜色立方体。

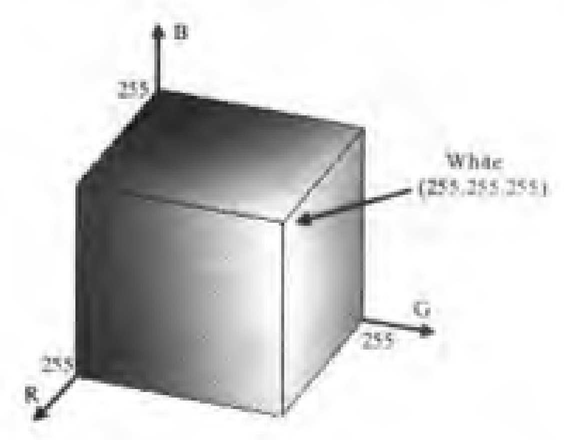


图 19-1　uint8 型图像的 RGB 颜色立方体

量子化需要将 RGB 颜色立方体分解成许多更小的小立方体，然后将落在每个小立方体中的所有颜色映射到小立方体的中心。均匀量子化和最小方差量子化的区别在于分解颜色立方体的方法不同。使用均匀量子化，颜色立方体被分解成相同大小的小立方体。使用最小方差量子化，颜色立方体被分解为不同大小的小立方体，小立方体的大小与图像中颜色的分布有关。

① 均匀量子化

调用 rgb2ind 函数并指定一个阈值，进行均匀量子化。阈值决定小立方体的大小，其进行设置的允许范围为[0, 1]。例如，如果指定一个阈值 0.1，则小立方体的边是 RGB 颜色立方体长度的十分之一，并且小立方体的最大总个数为

```
n = (floor(1/tol)+1)^3
```

下面的命令用阈值 0.1 进行均匀量子化。

```
RGB = imread('peppers.png');
[x,map] = rgb2ind(RGB, 0.1);
```

图 19-2 演示了 uint8 型图像的均匀量子化。为了看得更清楚，图像显示了一个二维截面。该截面上红色分量为 0，绿色和蓝色分量从 0 变到 255。实际像素值是多个 x 构成的图形中心的像素值。

② 最小方差量子化

如果是进行最小方差量子化，调用 rgb2ind 函数并在输出图像的颜色查找表中指定最大颜色种数。指定的个数决定颜色立方体分解的小立方体的个数。下面的命令行用最小方差量子化方法创建一个具有 185 种颜色的索引图像。

```
RGB = imread('peppers.png');
[X,map] = rgb2ind(RGB,185);
```

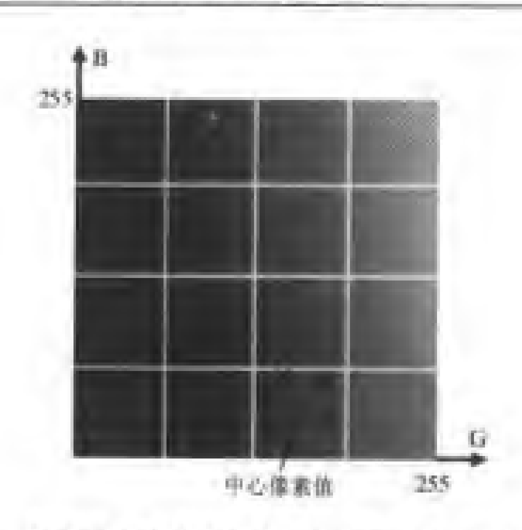


图 19-2　在 RGB 颜色立方体的一个剖面上进行均匀量子化

最小方差量子化基于像素值之间的方差将像素连成组。在最小方差量子化中，被分割出来的小立方体大小是变化的。如果颜色立方体的某些地方没有像素，那里就不会有小立方体。

设置小立方体的个数 *n* 时，小立方体的位置由分析图像中颜色数据的算法确定。一旦图像被分解为 *n* 个位置最优的小立方体，则每个立方体中的像素就会映射到小立方体中心的像素值上，就像均匀量子化那样。

最后生成的颜色查找表一般包含指定的入口个数。颜色立方体已经被分解以后，每个区域内至少要包含输入图像中的一种颜色。如果输入图像的颜色种类比指定的种数少，则输出颜色查找表中将少于 *n* 种颜色，并且输出图像将包含输入图像的所有颜色。

图 19-3 显示了与图 19-2 相同位置的颜色立方体一个剖面上的最小方差量子化。

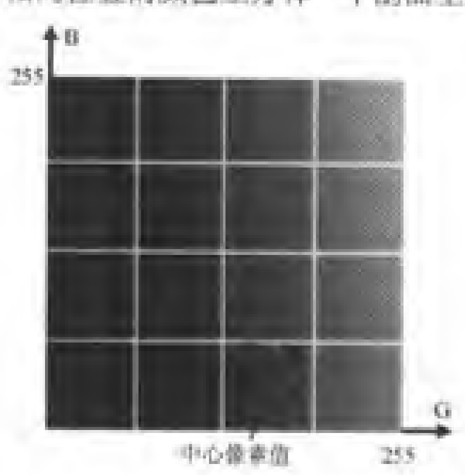


图 19-3　RGB 颜色立方体一个剖面上的最小方差量子化

对于给定的颜色数目，最小方差量子化能比均匀量子化获得更好的结果，因为它考虑了实际数据。

（2）颜色查找表映射

如果指定了要使用的实际颜色查找表，rgb2ind 函数会用颜色查找表映射代替量子化在指定的颜色查找表中查找颜色。如果需要创建使用固定颜色查找表的图像，这个方法是很有用

的。例如，需要基于 8 位模式显示多幅索引图像，可以将它们全部映射到同一个颜色查找表。如果指定的颜色查找表包含与 RGB 图像中相似的颜色，则颜色映射可以获得好的近似。如果颜色查找表不包含与 RGB 图像中相似的颜色，这个方法得到的结果将会是糟糕的。

下面的例子演示如何将两幅图像映射到同一颜色查找表。用 colorcube 函数创建该颜色查找表。colorcube 函数生成一个包含指定颜色数目的 RGB 颜色查找表。因为颜色查找表包括 RGB 颜色立方体中的所有颜色，输出图像理所当然地会与输入图像近似。

```
RGB1 = imread('autumn.tif');
RGB2 = imread('peppers.png');
X1 = rgb2ind(RGB1,colorcube(128));
X2 = rgb2ind(RGB2,colorcube(128));
```

2. 在索引图像中进行减色

需要在索引图像中减少颜色数目时，使用 imapprox 函数。该函数基于 rgb2ind 函数并且使用同一近似方法。实际上，imapprox 函数首先调用 ind2rgb 函数，把图像转换为 RGB 格式，然后调用 rgb2ind 函数返回一个具有更少颜色的新的索引图像。

例如，下面的命令行创建一个具有 64 种颜色的 trees 图像，其原始图像为 128 色。

```
load trees
[Y,newmap] = imapprox(X,map,64);
imshow(Y, newmap);
```

最终生成的图像的质量与近似计算所采用的方法、输入图像中颜色的范围和是否使用抖动有关。

3. 抖动

使用 rgb2ind 或 imapprox 函数减少图像中颜色的数目时，生成的图像看起来质量可能要比原始图像差一些，因为有些颜色已经没有了。rgb2ind 和 imapprox 函数都用抖动来增加输出图像中外观上的颜色个数。抖动改变邻域内像素的颜色，这样，每个邻域内的平均色与原始 RGB 颜色近似。

下面结合一个例子来介绍抖动的工作原理。假设有一幅包含很多深橙色像素的图像，这些深橙色在颜色查找表中没有精确匹配的颜色。为了生成这种橙色外观，图像处理工具箱从颜色查找表中选择了一套颜色组合，把它们放在一起，作为一个 6 像素组近似所需要的粉红色阴影。从远处看，这些像素能够产生比较理想的效果，但是凑近看，可以发现它是其他颜色混合起来的。下面的命令行载入一幅 24 位的图像，然后用 rgb2ind 函数创建两幅只有 8 种颜色的索引图像。

```
rgb=imread('onion.png');
imshow(rgb);
[X_no_dither,map]=rgb2ind(rgb,8,'nodither');
[X_dither,map]=rgb2ind(rgb,8,'dither');
figure, imshow(X_no_dither,map);
figure, imshow(X_dither,map);
```

生成图 19-4 中的 3 幅图。其中图（a）为原始图像，图（b）为没有抖动的图像，图（c）为抖动后的图像。



图 19-4　图像的抖动

可以发现，抖动后的图像有大量的表面颜色，但是有些模糊。没有抖动的图像表面颜色更少，但是与抖动后的图像相比更清晰。减色以后不进行抖动可能带来的问题是新图像会包含错误的轮廓。

19.2　坐标系统

根据上下文的不同，图像中的位置可以用不同的坐标系统来表达。本节讨论图像处理工具箱中用到的两个主要的坐标系统和它们之间的关系。这两个坐标系统分别是像素坐标系统和空间坐标系统。

19.2.1　像素坐标

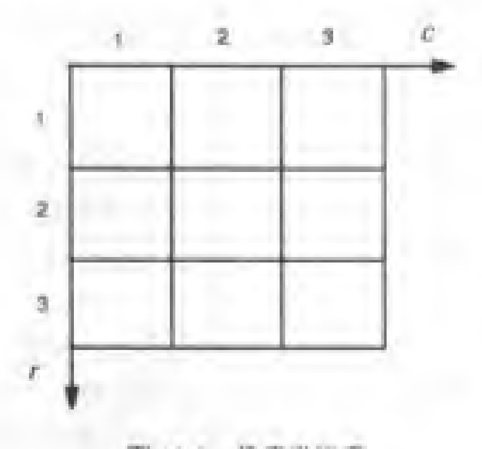


图 19-5　像素坐标系

通常，在图像中表示位置的最方便的方法是使用像素坐标。在这个坐标系统中，图像被看作一个离散元素形成的网格，并且按从上到下从左到右的顺序排列，如图 19-5 所示。

对于像素坐标，r 轴（行）向下为正，c 轴（列）向右为正。像素坐标为整型值，并且界于 1 和行或列的长度之间。

在像素坐标和 MATLAB 用于表示矩阵脚标的坐标之间有一个一一对应的关系。这种对应关系使得图像的数据矩阵和图像显示方式之间的关系更易于理解。例如，第 5 行第 2 列像素上的数据保存在矩阵元素（5, 2）上。

19.2.2　空间坐标

在像素坐标系统中，像素被认为是一个离散的单元，由一个单一的坐标对（如（5,2））惟一识别。从这一点上来说，一个类似（5.3, 2.2）的位置是没有意义的。

但是，有时候把像素当作方形的小块是有益的。这时候，（5.3, 2.2）是有意义的，并且与（5, 2）区分开来。在这个空间坐标系统中，图像中的位置是平面位置，用 x 和 y 表示，而不是像在像素坐标系统中那样用 r 和 c 表示。

图 19-6 表示用于图像的空间坐标系统。注意，y 轴向下为正。

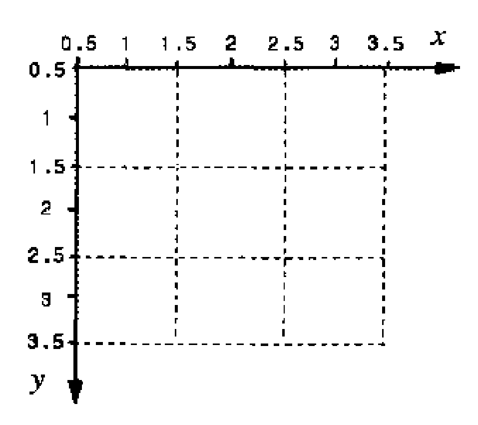


图 19-6　空间坐标系

在某些方面，空间坐标系统与像素坐标系统紧密对应。例如，任何像素中心点的空间坐标与该像素的像素坐标相同。

但是，也存在一些很重要的差别。在像素坐标系统中，图像的左上角坐标为（1, 1）；而空间坐标系统中，默认时该位置的坐标为（0.5, 0.5）。这是因为像素坐标系统是离散的，而空间坐标系统是连续的。还有，在像素坐标系统中，左上角总是（1, 1），但是在空间坐标系统中，可以指定非默认的原点。

另一个容易混淆的区别在很大程度上是一个约定问题：这两个系统中，表示水平向和垂向位置的数值前后位置是颠倒的。如前所述，像素坐标表示为（*r*, *c*），而空间坐标表示为（*x*, *y*）。后面的叙述中，如果函数语法使用 *r* 和 *c*，说明使用的是像素坐标；如果语法使用的是 *x* 和 *y*，说明使用的是空间坐标。

默认时，图像的空间坐标与像素坐标相对应。例如，第 5 行第 3 列的像素的中心点的空间坐标是 x=3，y=5。（注意，坐标的次序颠倒了。）这种对应关系在很大程度上简化了许多工具箱函数。有些函数主要使用空间坐标而不是像素坐标，但是，在使用默认的空间坐标系统的时候，可以指定像素坐标系统中的位置。

但是，在某些情况下，可能希望使用非默认的坐标系统。例如，可能希望把（19.0, 7.5）指定为图像的原点。如果调用一个函数返回图像的坐标，则返回的坐标将是非默认空间坐标系统中的值。

要建立一个非默认的空间坐标系统，可以在显示图像时指定 XData 和 YData 图像属性。这些属性是控制图像坐标范围的二元素向量。默认时，对于图像 A，XData 是[1 size(A,2)]，YData 是[1 size(A,1)]。

例如，如果 A 是一个 100 行 200 列的图像，默认的 XData 值为[1 200]，默认的 YData 值为[1 100]。这些向量的值实际上是第一个像素和最后一个像素中心点的坐标，所以，实际的坐标范围要略大一点。例如，如果 XData 是[1 200]，则 *x* 轴范围是[0.5 200.5]。

下面的命令用非默认的 XData 和 YData 显示图像。

```
A = magic(5);
x = [19.5 23.5];
y = [8.0 12.0];
image(A,'XData',x,'YData',y), axis image, colormap(jet(25))
```

生成图 19-7。

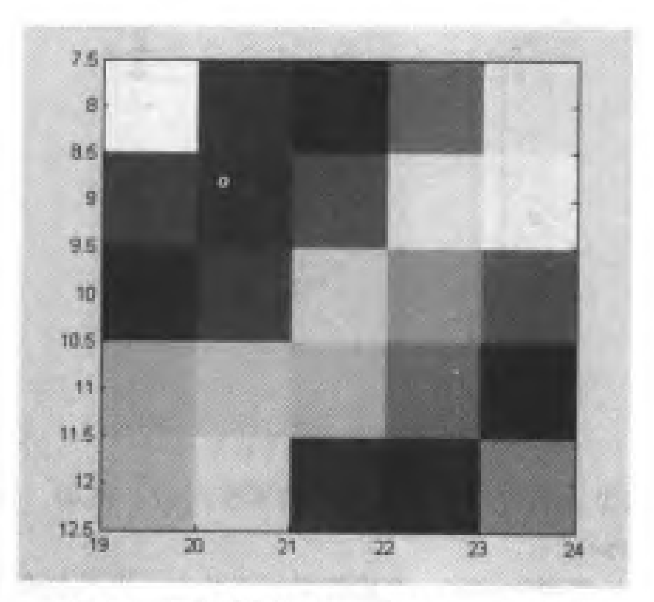


图 19-7　用非默认的 XData 和 YData 显示图像

第 20 章 图 像 合 成

图像合成又称为图像融合，包括图像的代数运算和逻辑运算等内容。图像处理工具箱提供了专门的图像代数运算函数，采用 MATLAB 的逻辑操作符，可以对二值图像进行逻辑运算。

20.1 代数运算

图像的代数运算是基于图像的标准算术运算（如加、减、乘、除）。无论是作为复杂图像处理操作的前期处理还是作为运算本身，图像代数运算都有很多用途。例如，图像减运算可以用于区分同一场景或对象的两幅或多幅图像。

可以用 MATLAB 算术运算符完成图像运算。但是，使用这些运算符时，必须先将图像转换为 double 型。为了使图像算术运算更方便，图像处理工具箱包含了一系列实现所有数值型非稀疏数据的处理函数。使用这些函数的好处有：

- 不需要将数据转换为 double 型的操作。函数接受任何数值数据类型，如 uint8, uint16 和 double 等，并以相同格式返回生成的图像。注意，这些函数进行双精度运算，是逐元素进行的，但是在 MATLAB 工作空间中不会将图像转换为双精度值。
- 自动进行溢出控制。函数对返回值进行截断处理，使之适合相应的数据类型。

表 20-1 列出了工具箱提供的图像运算函数。

表 20-1 图像运算函数

函 数	描 述
imabsdiff	两幅图像的绝对差
imadd	两幅图像的和运算
imcomplement	图像的补运算
imdivide	两幅图像的除运算
imlincomb	计算两幅图像的线性组合
immultiply	两幅图像的积
imsubtract	两幅图像的差

需要注意的是，整型运算很容易发生溢出。例如，uint8 型数据能保存的最大值为 255，运算结果超出 255 将发生溢出。算术运算还会生成分数值，它不能用整型数组表示。

图像运算函数进行整型运算时使用下面的规则：

- 超过整型类型对应范围的值被自动截断。
- 分数值进行圆整。

例如，如果数据类型是 uint8 型，结果大于 255（包括 Inf），则设置为 255。表 20-2 列出了其他一些示例。

表 20-2　截断规则示例

结　果	类　型	截　断　值
300	uint8	255
-45	uint8	0
10.5	uint8	11

20.1.1　图像加运算

两幅图像相加或常数与图像相加，使用 imadd 函数。该函数将两幅图像的对应像素的值相加，将和返回给输出图像的对应像素。在图像处理中，图像加运算有很多用途。例如，下面的代码片段用加运算将一幅图像叠加到另一幅图像上。注意，两幅图像的大小和类型必须是相同的。

```
I = imread('rice.png');
J = imread('cameraman.tif');
K = imadd(I,J);
imshow(K)
```

生成图 20-1。



图 20-1　两幅图像的加运算结果

给图像的每个像素添加一个常数值，可以提高图像的亮度。例如，下面的图像加亮一幅 RGB 图像。

```
RGB = imread('peppers.png');
RGB2 = imadd(RGB, 50);
subplot(1,2,1); imshow(RGB);
subplot(1,2,2); imshow(RGB2);
```

生成图 20-2。其中左图和右图分别为加亮前后的图像。





图 20-2　加亮图像

对两幅图像进行加运算时，结果很容易发生溢出，uint8 型数据尤其如此。发生溢出时，imadd 函数将结果截断为数据类型所支持的最大值。为了避免出现这种现象，在进行加运算以前，有必要将图像转换为一个更大的数据类型，如 uint16 型。

20.1.2　图像减运算

用 imsubtract 函数完成图像与图像或常数的减运算。该函数将两幅输入图像的对应像素进行值的减运算，然后将结果返回给输出图像的对应像素。图像减运算可以作为复杂图像处理的前期操作。例如，可以用图像减运算探察同一场景的多幅图像的变化。下面的代码片段将背景取为当前图像的形态学开运算结果，然后从当前图像中减去该背景。

```
rice= imread('rice.png');
background = imopen(rice, strel('disk',15));
rice2 = imsubtract(rice,background);
imshow(rice),figure,imshow(rice2);
```

生成图 20-3。


(a)


(b)

图 20-3　图像减运算

图像减运算可能导致某些像素的值为负。对于无符号数据类型如 uint8 或 uint16，发生此类问题时，imsubtract 函数将负值截断为 0，显示效果为黑色。使用 imabsdiff 函数可以避免出现负值，并且保留这些像素值的差异。该函数计算两幅图像中对应像素的值之间的绝对值，所以结果总是非负的。

20.1.3　图像乘运算

两幅图像相乘，使用 immultiply 函数。该函数对两幅输入图像的对应像素进行逐元素的点乘（.*）运算，并且将计算结果返回给输出图像的对应像素。

图像与常数相乘是一个常见的图像处理操作。如果常数大于 1，结果图像变亮；如果常数小于 1，结果图像变暗。一般情况下，图像乘运算比图像加运算得到的明暗效果更自然，因为它更好地保留了图像之间的相对差异。例如，下面的代码用一个常数因子进行乘运算。

```
I = imread('moon.tif');
J = immultiply(I,1.2);
imshow(I);
figure, imshow(J)
```

生成图 20-4，其中图（a）和图（b）是进行乘运算前、后的图像。



(a)



(b)

图 20-4　图像乘运算

uint8 型图像相乘常常发生溢出。发生溢出时，immultiply 函数将结果截断为数据类型的最大值。为了避免截断，在进行乘运算以前，可以将 uint8 型图像转换为更大的数据类型，如 uint16 型。

20.1.4　图像除运算

使用 imdivide 函数对两幅图像进行除运算。该函数对输入图像的对应像素进行逐元素的除运算。immultiply 函数将结果返回给输出图像的对应像素。

与减运算类似，图像除运算可用于检测两幅图像之间的差异。但是，除运算不是给出每个像素之间的绝对差异，而是给出对应像素值的分数差异或比率。例如，下面的代码将背景取为当前图像的形态学开运算结果，然后用当前图像除以该背景。

```
I = imread('rice.png');
background = imopen(I, strel('disk',15));
Ip = imdivide(I,background);
imshow(Ip,[])
```

生成图 20-5。



图 20-5　图像除运算结果

20.1.5　嵌套调用图像运算函数

可以组合使用图像运算函数来完成一系列的操作。例如，为了计算两幅图像 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的均值 $\boldsymbol{C}$，即

$$C = \frac{A + B}{2}$$

输入

```
I = imread('rice.png');
I2 = imread('cameraman.tif');
K = imdivide(imadd(I,I2), 2);
```

与 uint8 或 uint16 型数据一起使用时，每个运算函数会在将结果传递给下一个运算以前进行截断操作。截断操作会显著地减少输出图像的信息量。完成这个系列运算的一个更好的方法是使用 imlincomb 函数，imlincomb 函数在双精度上完成所有中间过程的运算，只截断最终结果。

```
K = imlincomb(.5,I,.5,I2);
```

20.2 逻辑运算

对于二值图像，可以用 MATLAB 的逻辑操作符进行逻辑运算。下面结合两幅图像进行演示。

（1）首先读入两幅灰度图像 rice.png 和 cameraman.tif，将它们转换为二值图像并进行显示。在命令窗口输入下面的代码。

```
I = imread('rice.png');
J = imread('cameraman.tif');
I2=im2bw(I,0.4)
J2=im2bw(J,0.4);
imshow(I2)
figure;imshow(J2)
```

生成图 20-6 和图 20-7，分别为原灰度图的二值图像。

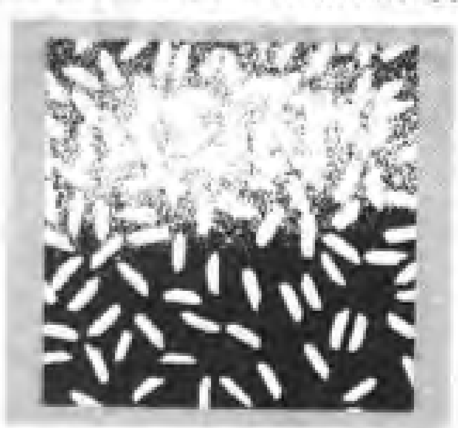


图 20-6 rice.png 的二值图像



图 20-7 cameraman.tif 的二值图像

用逻辑操作符对上面两幅图像进行与运算和或运算

```
imshow(I2 & J2)
figure;imshow(I2 | J2)
```

结果如图 20-8 和图 20-9 所示。

在命令窗口键入下面的命令行，进行非运算和异或运算。

```
figure;imshow(~J2)
figure;imshow(XOR(I2,J2))
```

结果如图 20-10 和图 20-11 所示。


图 20-8　与运算结果


图 20-9　或运算结果


图 20-10　非运算结果


图 20-11　异或运算结果

第21章　空间变换

图像的空间变换，或者说几何变换，指的是通过一定的几何运算，将图像经过平移、旋转、错切、缩放等变换操作以后显示在新的位置。工具箱中提供了部分函数实现图像的空间变换。

21.1　插值

插值用于估计图像上两个像素之间某个位置上的像素值。例如，放大一幅图像，则图像会比原始图像包含更多的像素，工具箱通过插值来获取其他像素的值。imresize 和 imrotate 函数使用了二维插值。

图像处理工具箱提供了 3 种插值方法：

- 最近邻插值；
- 双线性插值；
- 双三次插值。

几种插值方法的原理基本相同。每次处理时，首先在输入图像中找到与输出图像中对应的点，然后计算点附近某些序列的像素值的加权平均值，并将它赋给输出像素。权重由每个像素与点之间的距离确定。

3 种方法之间的区别主要在于点周围像素序列的取法不同。即

- 对于最近邻插值，输出像素的值指定为点所属像素的值，不考虑其他像素。
- 对于双线性插值，输出像素的值是最近的 2×2 邻域内像素值的加权平均值。
- 对于双三次插值，输出像素的值是最近的 4×4 邻域内像素值的加权平均值。

参与计算的像素的个数会影响计算的复杂度。所以，双线性插值法花费的时间比最近邻法的要长一些，而双三次法花费的时间比双线性法的又要长一些。但是，参与计算的像素越多，计算结果越精确。所以，在计算时间与质量之间有一个折中问题。

使用插值的函数有一个指定插值方法的变量。对于大部分这样的函数，默认的使用方法是最近邻法。该方法对于大部分图像类型生成可以接受的结果，而且它是对索引图像也合适的惟一方法。但是，对于亮度图像和 RGB 图像，通常应该指定双线性插值或双三次插值，因为这两种方法比最近邻法得到的结果更好。

对于 RGB 图像，插值计算在红色、绿色和蓝色平面上是分别进行的。

对于二值图像，插值能产生可以察觉的效果。如果使用双线性法或双三次法插值，则输出图像中像素的计算值将不会全部是 0 或 1。最终输出的图像效果与输入图像的类型有关。即

- 如果输入图像是 double 型的，则输出图像是 double 型的灰度图像。输出图像不是二值图像，因为它包括了 0 和 1 以外的值。

● 如果输入图像是 uint8 型的，则输出图像是 uint8 型的二值图像。插入的像素值会圆整到 0 或 1，所以输出图像可以是 uint8 型的。

如果使用最近邻插值，结果将总是二值图像，因为插入像素的值是直接取自输入图像的。

21.2 图像缩放

用 imresize 函数改变图像的大小。使用该函数，可以指定输出图像的大小、插值的方法和用于防止出现走样的滤波器。

21.2.1 指定输出图像的大小

使用 imresize 函数，可以用两种方法指定输出图像的大小：

- 指定放大倍率；
- 指定输出图像的尺寸。

1．指定放大倍率

指定一个大于 1 或者大于 0 小于 1 的数，可以放大或缩小图像。例如，下面的命令把图像大小放大到原来的 1.25 倍。

```
I = imread('circuit.tif');
J = imresize(I,1.25);
imshow(I)
figure, imshow(J)
```

生成图 21-1，其中图（a）和图（b）分别为放大前后的图像。



（a）



（b）

图 21-1　放大前后的图像

2．指定输出图像的尺寸

可以通过传递一个包含输出图像行数和列数的向量来指定输出图像的大小。下面的命令创建一个输出图像 Y，它有 100 行 150 列。

```
Y = imresize(X,[100 150])
```

21.2.2 指定插值方法

默认时，imresize 函数使用最近邻插值法来确定输出图像中的像素值，但是可以指定其他的插值方法。表 21-1 按照复杂性列出了工具箱所支持的插值方法。

表 21-1 工具箱支持的插值方法

变量值	插值方法
'nearest'	最近邻插值法（默认方法）
'bilinear'	双线性插值
'bicubic'	双三次插值

本例中，imresize 函数使用了双线性插值法。

```
Y = imresize(X,[100 150],'bilinear')
```

21.2.3 用滤波器防止走样

缩小图像会导致输出图像中出现一些人为失真的现象，比如走样，因为缩小图像时总会丢失一些信息。输出图像中，走样看起来像波纹。

用双线性或双三次插值的方法缩小图像时，imresize 函数会在插值以前自动对图像使用低通滤波器来降低走样产生的影响。可以指定滤波器的大小或者指定不同的滤波器。

注意，即使使用低通滤波器也会造成人为失真，因为缩小图像时总会丢失信息。

如果使用了最近邻插值，imresize 函数不会采用低通滤波器。最近邻插值只适用于索引图像，而低通滤波不适合这些图像。

21.3 旋转图像

用 imrotate 函数旋转图像。该函数接受两个主要的变量，即要旋转的图像和旋转角度。旋转角度的单位为度。如果指定一个正值，imrotate 函数按逆时针方向旋转图像；如果指定一个负值，imrotate 函数按顺时针方向旋转图像。下面的例子将图像 I 逆时针方向旋转 35 度。

```
J = imrotate(I,35);
```

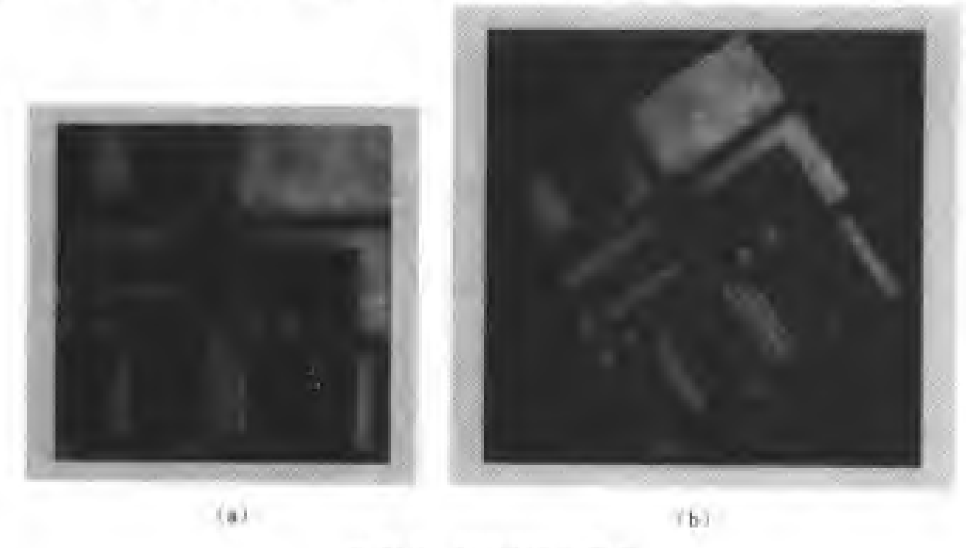
作为可选变量，还可以给 imrotate 函数指定插值方法和图像的大小。

21.3.1 指定插值方法

默认时，imrotate 函数使用最近邻插值法确定输出图像中像素的值，但是也可以指定其他方法。其他方法有双线性插值法和双三次插值法。例如，下面的代码采用双线性插值法，将图像逆时针方向旋转 35 度。

```
I = imread('circuit.tif');
J = imrotate(I,35,'bilinear');
imshow(I)
figure, imshow(J)
```

生成图 21-2，其中图（a）和图（b）分别表示旋转前后的图像。



（a）　　（b）

图 21-2　旋转前后的图像

21.3.2　指定输出图像的大小

默认时，imrotate 函数创建一个足够包含整个原始图像的输出图像。落在原始图像边界以外的像素值设置为 0，并且在输出图像中显示为黑色的背景。如果把文本字符串 "crop" 指定为变量，则 imrotate 函数会将输出图像裁剪成与输入图像的大小相同。

21.4　图像裁剪

用 imcrop 函数对图像进行裁剪。该函数接受两个主要变量，即要裁减的图像和定义裁剪区域的矩形坐标。

如果调用 imcrop 函数时没有指定裁剪矩形，可以交互式指定裁剪矩形。此时，当鼠标光标落在图像上方时，会变成十字形。把光标放在裁剪区域的一个角上，按下鼠标左键，然后拖拉至裁剪区域的对角。imcrop 函数会在所选区域的周围画一个矩形。释放鼠标时，imcrop 函数从选择区域创建一个新图像。

下面的例子显示一幅图像并调用 imcrop 函数。该函数把图像显示在一个图像窗口中，并且等待你在图像上绘裁剪矩形。图中，选择的矩形显示为红色。然后调用 imshow 函数查看裁剪后的图像。

```
imshow circuit.tif
I = imcrop;
imshow(I);
```

生成图 21-3。在图中用鼠标绘红线圈出来的矩形框，裁剪出如图 21-4 所示的图像部分。



图 21-3 在原图像上选择要裁剪的部分



图 21-4 裁剪出的图像

21.5 进行一般的空间变换

用 imtransform 函数完成一般的二维空间变换。该函数接受两个主要变量，即要变换的图像和一个称为 TFORM 的空间变换结构，该结构指定变换类型。

在 TFORM 结构中指定变换类型，创建一个 TFORM 结构有两种方法，即使用 maketform 函数和 cp2tform 函数。

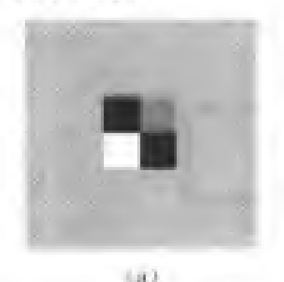
使用 maketform 函数时，可以指定变换类型。表 21-2 按字母先后顺序列出了各种变换类型。

表 21-2 各种变换类型

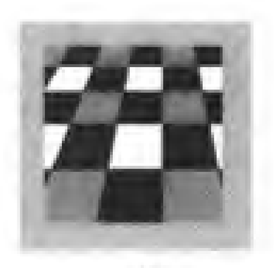
变换	描述
'affine'	变换，包括平移、旋转、比例、拉伸和错切等。直线仍为直线，平行线仍为平行线，但矩形可能变成平行四边形
'box'	是 affine 变换的特例
'composite'	两种或两种以上变换的组合
'custom'	自定义变换
'projective'	投影变换

一旦在 TFORM 结构中定义了变换，就可以通过调用 imtransform 函数进行变换操作。例如，下面的代码用 imtransform 函数进行跳棋图像的投影变换。

```
I = checkerboard(20,1,1);
figure; imshow(I)
T = maketform('projective',[1 1; 41 1; 41 41;    1 41],...
[5 5; 40 5; 35 30; -10 30]);
R = makeresampler('cubic','circular');
K = imtransform(I,T,R,'Size',[100 100],'XYScale',1);
figure, imshow(K)
```



(a)



(b)

图 21-5 投影变换前后的图像

生成图 21-5，其中图（a）和图（b）分别为进行投影变换前后的图像。

第 22 章 邻域和块处理

22.1 块处理操作

有些图像处理操作是逐块处理的，而不是一次处理整幅图像。图像处理工具箱提供了几个函数来进行块操作，例如，进行图像膨胀的 imdilate 函数。另外，工具箱提供了进行图像块处理的更一般的函数。下面介绍这些一般的块处理函数。

使用这些函数中的任何一个，需要提供与块大小有关的信息，并单独指定一个函数来处理块。这个单独指定的函数将输入图像分成不同的块，为每个块调用指定的函数并将结果重新分配给输出图像。

使用这些函数，可以完成不同的块处理操作，包括滑动邻域操作和分离块操作。

- 进行滑动邻域操作时，输入图像是逐像素进行处理的。即，对于输入图像中的每个像素，进行某些操作来确定输出图像中对应像素的值。操作基于相邻像素块的值。
- 进行分离块操作时，输入图像是逐块进行操作的。即，图像分成几个矩形块，并且有些操作是单独对每个块进行的，以便确定输出图像的对应块中像素的值。

另外，工具箱提供了进行列处理操作的函数。这些操作与块操作没有实质性的区别，不仅如此，它们还通过将块重置到一个矩阵列来加速块操作。

22.2 滑动邻域操作

滑动邻域操作每次处理一个像素，输出图像中任何给定像素的值都通过给输入图像中对应像素邻域内像素值应用一个算法来确定。某像素的邻域是指由该像素的相对位置确定的一系列像素。邻域是一个矩形块，在图像矩阵中从一个元素向下一个元素移动时，邻域块向相同方向滑动。

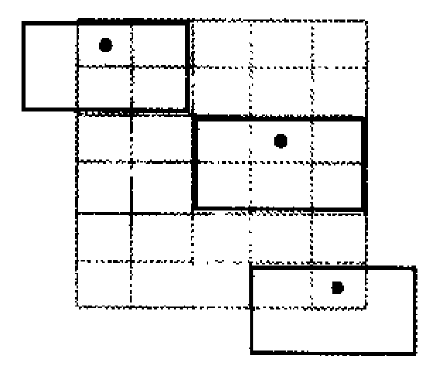


图 22-1 6×5 矩阵中的相邻块

图 22-1 显示了一个 2×3 的邻域块在 6×5 的矩阵中滑动的情况。每个邻域块的中心像素用圆点标注。

中心像素是输入图像中实际要处理的像素。如果邻域的行数和列数都是奇数，则中心像素位于邻域的中心。如果某个维的长度为偶数，则中心像素靠近位于中心的左侧或上侧。例如，在一个 2×2 的邻域中，中心像素是左上角的那个元素。

对于任何 $m\times n$ 的邻域，中心像素为

floor(([m n]+1)/2)

按照以下步骤进行滑动邻域操作。

（1）选择一个像素。

（2）确定这个像素的邻域。

（3）将一个函数应用于邻域中的像素值。这个函数必须返回一个标量。

（4）找到输出图像中的像素，它的位置对应于输入图像中中心像素的位置。将这个输出像素的值设置为函数的返回值。

（5）对于输入图像中的每个像素，重复步骤 1 至 4。

例如，该函数可能是一个求平均值的操作，即首先将邻域内像素的值加起来，然后除以邻域内的像素个数。计算结果就是输出像素的值。

邻域块在整个图像上滑动，邻域内的某些像素可能缺失，特别是中心像素位于图像边界上的时候。例如，如果中心像素是图像左上角的像素，则对应邻域会包含部分不属于图像的像素。

处理这些邻域时，滑动邻域操作通常用多个 0 来填充图像边界。换句话说，这些函数通过假设图像被额外的 0 组成的行和列包围来处理边界像素。这些行和列不会成为输出图像的一部分，并且只用于图像中实际像素的邻域的一部分。

可以用滑动邻域操作实现多种滤波。实例之一是卷积，它实现线性滤波。MATLAB 提供了 conv 和 filter2 函数，工具箱提供了 imfilter 函数进行卷积。

除了卷积以外，还有很多其他的滤波操作可以通过滑动邻域实现。这种操作实际上是非线性的，例如，可以在输出像素的值等于输入像素的邻域内像素值的标准差的地方实现滑动邻域操作。

可以用 nlfilter 函数实现多种滑动邻域操作。nlfilter 函数的输入变量有一幅图像、邻域大小和一个返回标量的函数，返回一幅大小与输入图像相同的图像。输出图像中每个像素的值通过将对应输入像素的邻域传递给返回标量的那个函数来进行计算。例如，下面的调用通过计算输入像素的 3×3 邻域内像素值的标准差来获得输出像素的值。

```
I2 = nlfilter(I,[3 3],'std2');
```

可以编写一个 M 文件来实现这个返回标量的函数，然后将它传递给 nlfilter 函数。例如，下面的命令用一个名为 myfun 的函数，按 2×3 的邻域来处理图像 I。

```
nlfilter(I,[2 3],@myfun);
```

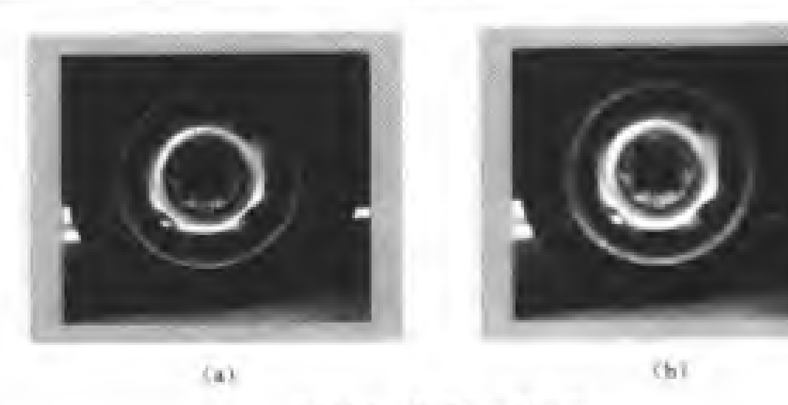
@myfun 使用了函数句柄，也可以用命令行函数，例如，

```
f = inline('sqrt(min(x(:)))');
I2 = nlfilter(I,[2 2],f);
```

下面的例子用 nlfilter 函数将每个像素的值设置为 3×3 邻域内的最大值。

```
I = imread('tire.tif');
f = inline('max(x(:))');
I2 = nlfilter(I,[3 3],f);
imshow(I);
figure, imshow(I2);
```

生成图 22-2，其中图（a）和图（b）分别为处理前后的图像。



(a)　　(b)

图 22-2　处理前后的图像

如果计算是针对矩阵列而不是矩形邻域进行的，则 nlfilter 函数可以快得多地实现许多操作。

22.3　分离块操作

分离块是将矩阵分成 $m \times n$ 部分的矩形分离框。分离块从图像的左上角开始无重叠地覆盖图像矩阵。如果这些块不能精确覆盖图像，则工具箱进行 0 填充。图 22-3 中将一个 15×30 的矩阵分离成 4×8 块。

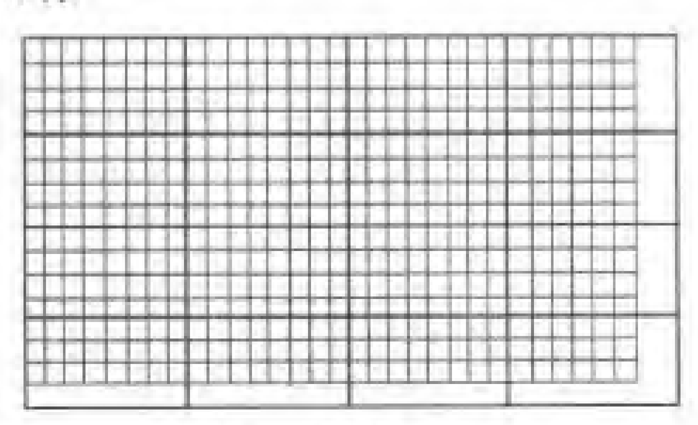


图 22-3　将图像分离成不同的块

0 填充处理在必要的时候将 0 添加到图像矩阵的底部和右侧。进行 0 填充以后，矩阵成为 16×32 的。

函数 blkproc 进行分离块操作。该函数从图像中提取分离块并将它传递给指定的函数，然后将返回的块集中起来形成输出图像。例如，下面的命令行用 myfun 函数处理块为 4×6 的图像 I。

```
I2 = blkproc(I,[4 6],@myfun);
```

将 myfun 函数写成如下的命令行的形式。

```
f =  inline('mean2(x)*ones(size(x))');
I2 = blkproc(I,[4 6],f);
```

下面的例子用 blkproc 函数将图像矩阵中每一个 8×8 的块中每个像素的值设置为该块

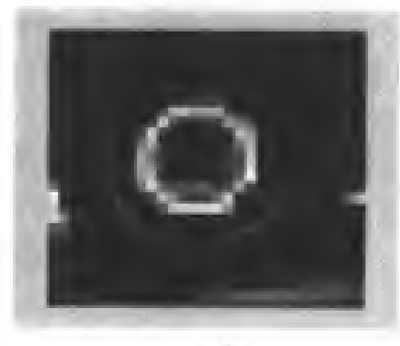
中所有元素的平均值。

```
I = imread('tire.tif');
f = inline('uint8(round(mean2(x)*ones(size(x))))');
I2 = blkproc(I,[8 8],f);
imshow(I)
figure, imshow(I2);
```

生成图 22-4，其中图（a）和图（b）分别为处理前后的图像。



（a）



（b）

图 22-4　处理前后的图像

调用 blkproc 函数定义分离块时，可以指定将这些块相互重叠，即，可以指定块外额外的像素行和列，处理块时将它们的值考虑进去。存在重叠时，blkproc 函数将扩展的块传递给指定的函数。

图 22-5 中显示了 15×30 矩阵中某些块进行 1×2 重叠以后形成的重叠区域。每个 4×8 的块上下都有一行的重叠，左右两侧都有两列的重叠。图中，阴影表示重叠。4×8 的块从左上角开始覆盖图像矩阵。

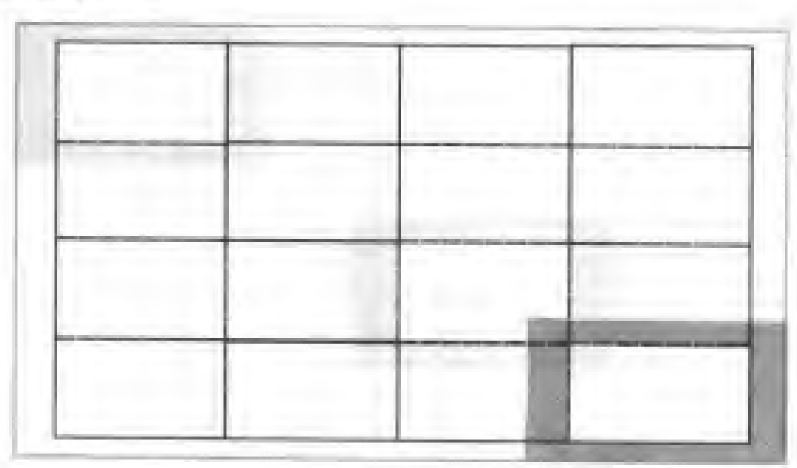


图 22-5　对矩阵中的某些块进行重叠

指定重叠，需要给 blkproc 函数另外提供一个输入变量。用函数 myfun 处理图 22-5 中的块，使用下面的调用形式。

```
B = blkproc(A,[4 8],[1 2],@myfun)
```

重叠常常会增加所需要的 0 填充的数量。例如，图 22-5 中，原来的 15×30 矩阵成了

有 0 填充的 16×32 矩阵。当这个 15×30 矩阵包含一个 1×2 的重叠时，填充后的矩阵成了 18×36 的矩阵。图像中最外面的矩阵描绘了填充以后图像的新边界。

22.4 列处理

工具箱提供了多个把滑动邻域或分离块作为列进行处理的函数，它们对于那些在 MATLAB 中按列处理的操作来说很有用。很多时候，列处理可以减少图像处理的运行时间。例如，假设正在进行的操作需要计算每个块的均值，则将这些块重置为列以后再进行计算要快得多，因为调用 mean 函数一次就可以计算每一列的均值，而不需要多次调用 mean 函数来单独计算每一个块的均值。

可以用 colfilt 函数进行列处理，该函数可以实现以下操作：

（1）将图像矩阵的每一个滑动块或分离块重塑为一个暂时矩阵的列；

（2）将这个暂时矩阵传递给一个指定函数；

（3）将生成的矩阵重置为原来的形状。

22.4.1 滑动邻域操作

对于滑动邻域操作，colfilt 函数创建一个矩阵，矩阵中的每一列对应于原始图像中的一个像素。列对应于一个给定像素，值为原图像中像素邻域的值。

图 22-6 中演示了这种处理方法。图中，一个 65 的图像矩阵按照 23 的块进行处理。colfilt 函数为图像中每个像素创建一个列，这样，暂时图像中一共有 30 列。每个像素对应的列包含了它邻域内像素的值，所以有 6 行。colfilt 函数在必要时对输入图像进行 0 填充。例如，图 22-6 中，由于 0 填充，左上角像素有两个 0 值。

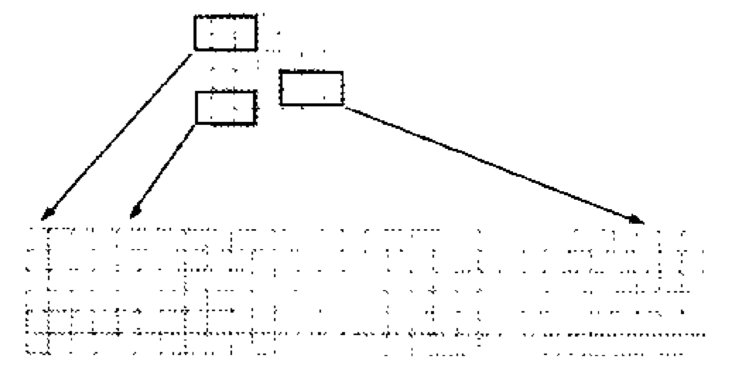


图 22-6　为滑动邻域操作创建一个暂时矩阵

暂时矩阵被传递给一个函数，该函数必须为每一列返回一个单一的值。然后，这个返回值指定给输出图像中合适的像素。

colfilt 函数可以用更少的运行时间，生成与 nlfilter 函数相同的结果。但是，它要使用更多的内存。下面的例子将每一个输出像素的值设置为输入像素邻域内的最大值。

```
I2 = colfilt(I,[3 3],'sliding',@max);
```

22.4.2　分离块操作

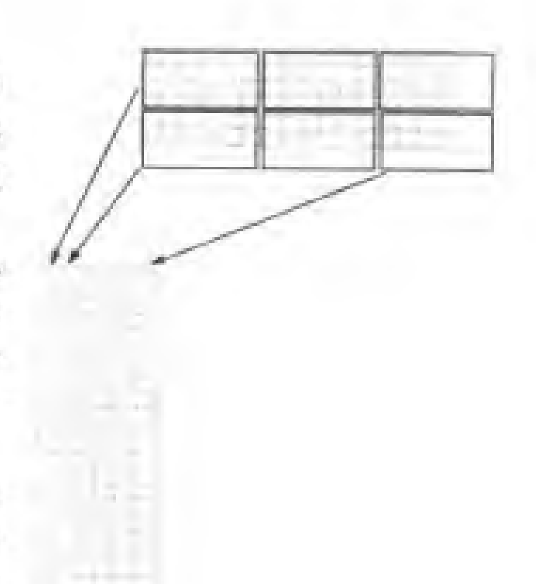


图 22-7　为分离块操作创建一个暂时矩阵

对于分离块操作，colfilt 函数通过将图像中的每一个块重置为列来创建暂时矩阵。如果必要，colfilt 函数在创建暂时矩阵以前用 0 填充原图像。

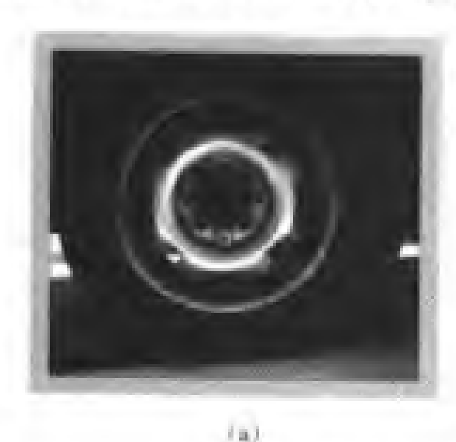
图 22-7 演示了这个处理方法。一个 6×16 的图像矩阵按照 4×6 的块进行了处理。colfilt 函数首先对图像进行 0 填充，使得原图像的大小变成 8×18，然后将块重置为 6 列，每一列 24 个元素。

将图像重置为暂时矩阵以后，colfilt 函数将该矩阵传递给一个必须返回与暂时矩阵大小相同矩阵的函数。如果块的大小是 $m\times n$，并且图像是 $mm\times nn$ 的，则暂时矩阵的大小为(m*n)×(ceil(mm/m)*ceil(nn/n))。函数处理暂时矩阵以后，输出重置为原图像矩阵的形状。

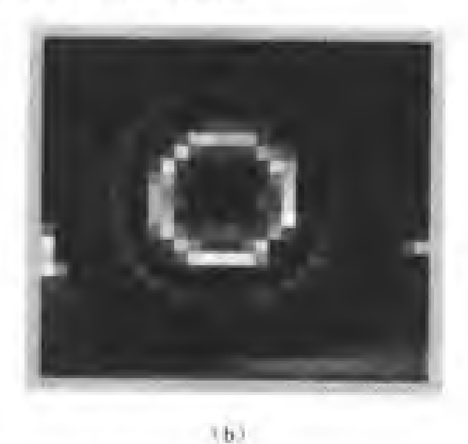
下面的例子将图像的每个 8×8 块中的所有像素的值设置为块中像素的平均值。

```
I = im2double(imread('tire.tif'));
f = inline('ones(64,1)*mean(x)');
I2 = colfilt(I,[8 8],'distinct',f);
imshow(I);figure;imshow(I2)
```

生成图 22-8，其中图（a）和图（b）分别为处理前后的图像。



(a)



(b)

图 22-8　处理前后的图像

第 23 章　线性滤波和滤波器设计

23.1　线性滤波

滤波是一门修改或增强图像的技术。例如，可以通过对一幅图像进行滤波来强调或删除图像的某些特征。滤波是一种邻域运算，即输出图像中任何像素的值都是通过采用一定的算法，根据输入图像中对应像素周围一定邻域内像素的值得到的。

线性滤波指的是输出像素的值是输入像素邻域内像素值的线性组合。下面介绍 MATLAB 和图像处理工具箱中采用的线性滤波器。包括：

- 用卷积和相关性描述滤波；
- 如何使用 imfilter 函数进行滤波；
- 关于使用预定义滤波器类型的讨论。

23.1.1　卷积

图像的线性滤波是通过一种称为卷积的运算来完成的。卷积时，输出像素的值是邻域内像素的加权和。权重矩阵称为卷积核，又称为滤波器。

例如，值设图像为

```
A = [ 17  24   1   8  15
      23   5   7  14  16
       4   6  13  20  22
      10  12  19  21   3
      11  18  25   2   9 ]
```

卷积核为

```
h = [ 8  1  6
      3  5  7
      4  9  2 ]
```

按照下面的步骤计算输出像素(2,4)的值：

（1）卷积核绕自己的核心元素旋转 180 度；

（2）移动卷积核的中心元素，使它位于 A 的元素(2,4)的上方；

（3）在旋转后的卷积核中，将 A 的像素值作为权重相乘；

（4）求得第 3 步各结果的和。

所以，输出像素(2,4)的值为

$$1\cdot 2+8\cdot 9+15\cdot 4+7\cdot 7+14\cdot 5+16\cdot 3+13\cdot 6+20\cdot 1+22\cdot 8=575$$

值的计算如图 23-1 所示。

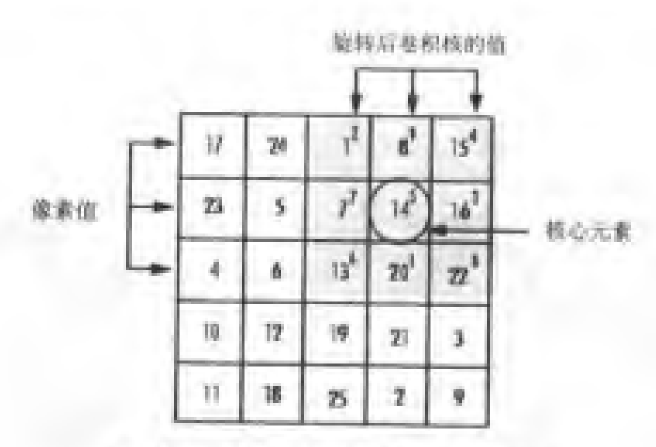


图 23-1 用卷积核计算像素(2,4)的值

23.1.2 相关性

相关性操作与卷积紧密相关。在相关性操作中，输出像素的值也作为相邻像素值的加权和进行计算。区别在于，在这里权重矩阵称为相关核，计算过程中不旋转。图 23-2 显示了如何计算 A 的相关性的输出像素(2,4)的值，假设 h 是相关核，按照下面的步骤进行操作：

- 移动相关核的中心元素到 A 的元素(2,4)的上方；
- 把 A 的像素值作为权重，乘以相关核；
- 将上面各步得到的结果相加。

由相关性操作得到的输出像素(2,4)的值为

18+8· 1+15· 6+7· 3+14· 5+16· 7+13· 4+20· 9=585

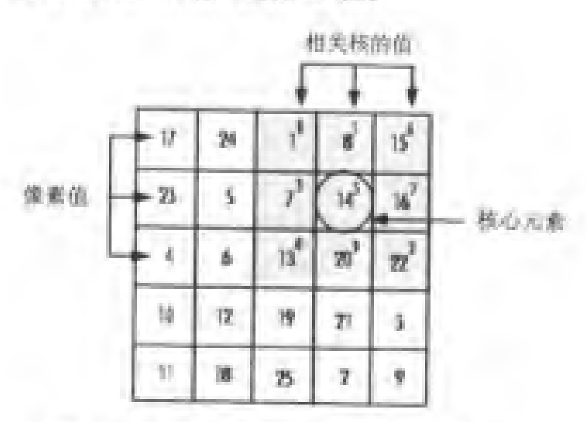


图 23-2 用相关操作计算像素(2,4)的值

23.1.3 用 imfilter 函数进行滤波

不管利用相关性还是卷积，用工具箱函数 imfilter 都可以进行图像滤波。下面的例子用一个包含相同权重的 5×5 的滤波器进行滤波。这一类滤波器常称为均值滤波器。

```
I = imread('coins.png');
h = ones(5,5) / 25;
```

```
I2 = imfilter(I,h);
imshow(I);
figure, imshow(I2)
```

滤波前后的图像如图 23-3 中图（a）和图（b）所示。



（a）



（b）

图 23-3 滤波前后的图像

1．数据类型

imfilter 函数操作数据类型的方式与图像算术运算函数的差不多，输出图像与输入图像具有相同的数据类型或数值类型。imfilter 函数采用双精度浮点值计算每个输出像素的值。如果结果超出了数据类型所限定的范围，则 imfilter 函数将结果数据截断到允许的范围。如果是整型数据类型，则 imfilter 函数对小数值进行圆整。

为了避开截断操作，可以在调用 imfilter 函数以前将图像转换为不同的数据类型。下面的例子中，当输入图像为 double 型时，imfilter 函数的输出含有负值。

```
A = magic(5)
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
h = [-1 0 1]
h =
    -1     0     1
imfilter(A,h)
ans =
    24   -16   -16    14    -8
     5   -16     9     9   -14
     6     9    14     9   -20
    12     9     9   -16   -21
    18    14   -16   -16    -2
```

下面假设 A 是 uint8 型的，则

```
A = uint8(magic(5));
imfilter(A,h)
ans =
    24     0     0    14     0
     5     0     9     9     0
     6     9    14     9     0
    12     9     9     0     0
    18    14     0     0     0
```

因为输入图像是 uint8 型的，所以输出图像也是 uint8 型的，而且负值全部截断为 0。此时，在调用 imfilter 函数前将图像转换为另一个数据类型如 single 型或 double 型等是合适的。

使用相关或卷积，imfilter 函数都可以进行滤波。默认时该函数使用相关。将字符串"conv"作为一个可选的输入变量传给 imfilter 函数，可以用卷积方法进行滤波。例如，

```
A = magic(5);
h = [-1 0 1]
imfilter(A,h)    % 使用相关进行滤波
ans =
24   -16   -16    14    -8
5    -16     9     9   -14
6      9    14     9   -20
12     9     9   -16   -21
18    14   -16   -16    -2
imfilter(A,h,'conv')    % 使用卷积进行滤波
ans =
   -24    16    16   -14     8
    -5    16    -9    -9    14
    -6    -9   -14    -9    20
   -12    -9    -9    16    21
   -18   -14    16    16     2
```

2. 边缘填充选项

计算图像边缘的输出像素值时，卷积或相关核的一部分通常位于图像边缘的外侧，如图 23-4 中所示。此时，imfilter 函数会假设这些位于图像边缘外侧的像素值为 0，这称为 0 填充，如图 23-5 所示。

图 23-4　核心元素的值落在图像外

图 23-5　0 填充

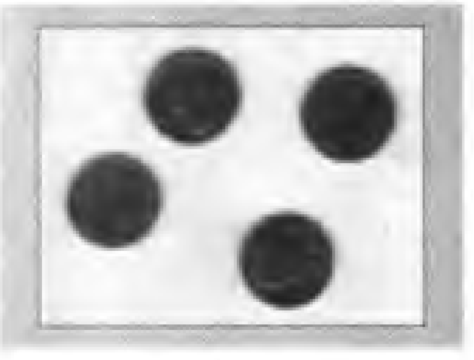
对图像滤波以后，常会沿图像边缘形成一条黑带。在命令窗口中输入下面的命令行，对图像进行滤波。

```
I = imread('eight.tif');
h = ones(5,5) / 25;
I2 = imfilter(I,h);
imshow(I);
figure, imshow(I2)
```

生成图 23-6，图（a）、（b）分别为滤波前后的图像。



（a）



（b）

图 23-6　滤波前后的图像

为了剔除 0 填充产生的黑带，imfilter 提供了一个替代的边界填充算法，称为边界复制。进行边界复制时，任何图像外侧的像素值取为距离最近的边缘像素的值，如图 23-7 所示。

使用边界复制法时，给 imfilter 函数传递一个额外的可选变量"replicate"。

```
I3 = imfilter(I,h,'replicate');
figure, imshow(I3);
```

生成图 23-8。

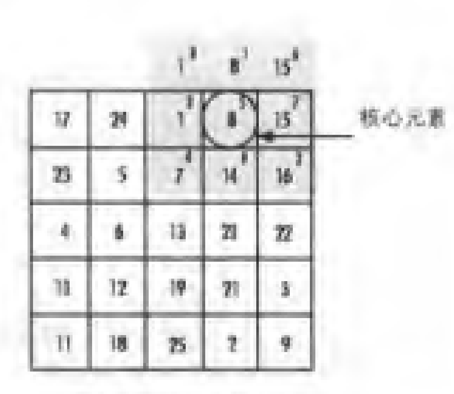


图 23-7　边界复制

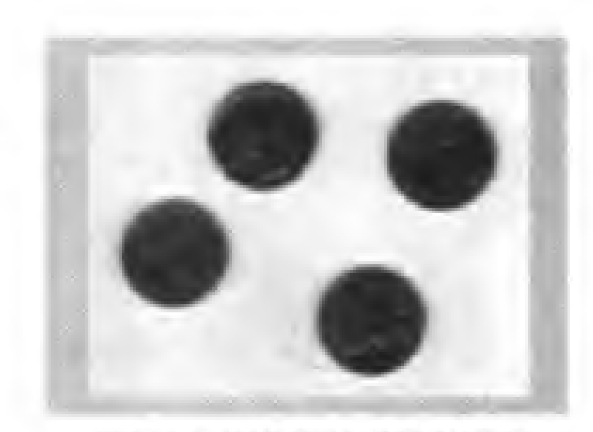


图 23-8　采用边界复制法得到的图像

imfilter 函数支持其他边界填充选项，如"circular"和"symmetric"等，可以参见 imfilter 函数的帮助文档。

3. 多维滤波

imfilter 函数可以处理多维图像和多维滤波器。滤波的一个方便之处在于，用二维滤波器对三维图像进行滤波等价于用同一个滤波器单独对三维图像的每一个面板进行滤波。例如，下面用同一个滤波器对一幅真彩色图像的每一个颜色面板进行滤波。

```
rgb = imread('peppers.png');
h = ones(5,5)/25;
rgb2 = imfilter(rgb,h);
imshow(rgb);
figure, imshow(rgb2);
```

生成图 23-9，图（a）、（b）分别为滤波前后的图像。



(a)



(b)

图 23-9　滤波前后的图像

MATLAB 中还有其他二维和多维滤波函数。函数 filter2 进行二维相关性处理，conv2 进行二维卷积处理，convn 进行多维卷积处理。这些函数总是将输入图像转换为 double 型的，输出图像也总是 double 型的。这些其他的滤波函数总是假设输入图像是 0 填充的，并且不支持其他填充选项。

23.1.4　使用预定义的滤波器类型

fspecial 函数用相关核的形式生成多种预定义滤波器。用 fspecial 函数创建一个滤波器以后，可以直接用 imfilter 函数将它用于图像数据中。下面的例子将一个滤波器用于灰度图像，使它的边缘和内部细节更清晰。

```
I = imread('moon.tif');
h = fspecial('unsharp');
I2 = imfilter(I,h);
imshow(I)
figure, imshow(I2)
```

生成图 23-10，图（a）、（b）分别为滤波前后的图像。



(a)　　(b)

图 23-10　滤波前后的图像

23.2　滤波器设计

下面介绍滤波器的设计。包括以下内容：

- FIR 滤波器；
- 将一维 FIR 滤波器转换为二维 FIR 滤波器的频率转换方法；
- 基于所需频率响应创建滤波器的频率取样方法；
- 通过将理想激励响应与窗口函数相乘来生成滤波器的窗口法；
- 创建所需的频率响应矩阵；
- 计算滤波器的频率响应。

23.2.1　FIR 滤波器

图像处理工具箱支持一种线性滤波器，即二维有限激励响应（FIR）滤波器。FIR 滤波器有几个特点，使得它在 MATLAB 环境下使用很理想：

- FIR 滤波器容易表示成系数矩阵；
- 二维 FIR 滤波器是一维 FIR 滤波器的自然拓展；
- 有几个有名的可靠方法可以用于 FIR 滤波器设计；
- FIR 滤波器容易实现；
- FIR 滤波器可以设计成具有线性阶段，它可以帮助防止变形。

23.2.2　频率变换方法

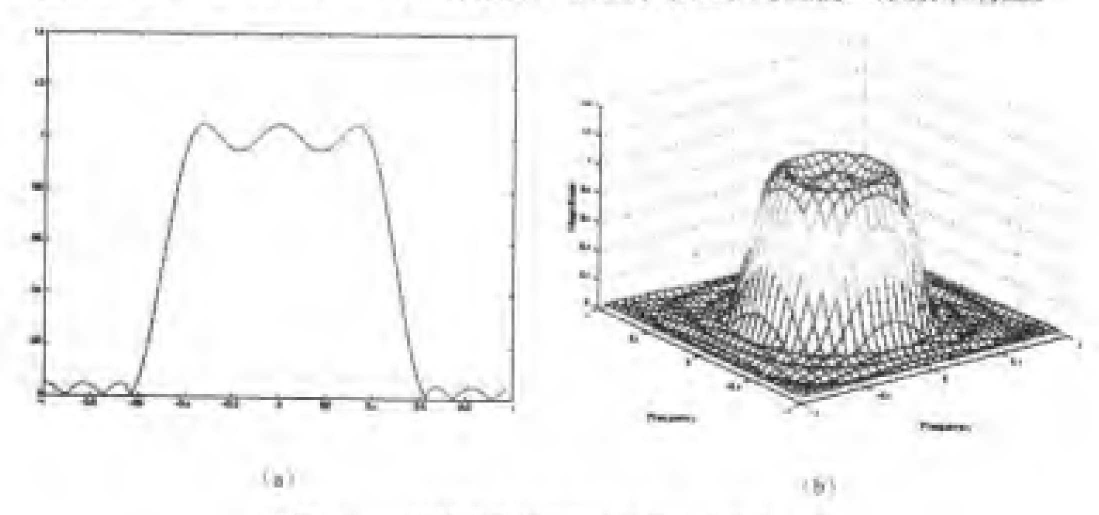
频率变换方法可以将一维 FIR 滤波器转换为二维 FIR 滤波器。频率变换方法可以保持一维 FIR 滤波器的大部分特点，特别是变换带宽和波的特征。该方法使用了变换矩阵，其元素定义频率变换。

工具箱函数 ftrans2 实现了频率变换方法。该函数的默认变换矩阵生成近于循环对称的

滤波器。通过定义自己的变换矩阵，可以获得不同的对称性。通常，频率变换法会生成很好的结果，因为生成一个特点鲜明的一维滤波器比生成一个对应的二维滤波器要容易些。例如，下面的例子设计一个一维 FIR 滤波器，然后用它创建一个性能相近的二维滤波器。

```
b = remez(10,[0 0.4 0.6 1],[1 1 0 0]);
h = ftrans2(b);
[H,w] = freqz(b,1,64,'whole');
colormap(jet(64))
plot(w/pi-1,fftshift(abs(H)))
figure, freqz2(h,[32 32])
```

生成图 23-11，其中图（a）和图（b）分别为一维频率响应和对应的二维频率响应。



（a）　　（b）

图 23-11　一维频率响应及其对应的二维频率响应

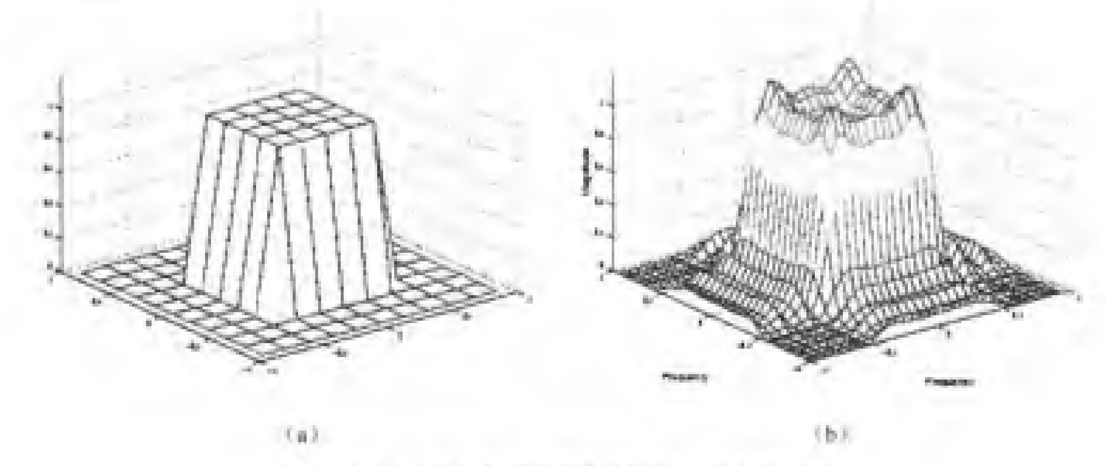
23.2.3　频率取样法

频率取样法创建一个基于所需频率响应的滤波器，给定定义频率响应形状的点的矩阵。该方法创建一个滤波器，该滤波器的频率响应通过这些点进行传递。频率取样对给定点之间的频率响应行为没有限制。

工具箱函数 fsamp2 实现了二维 FIR 滤波器的频率取样设计。fsamp2 函数返回一个滤波器 h，该滤波器有一个频率响应在输入矩阵 ***Hd*** 的各个点中传递。下面的例子用 fsamp2 函数创建一个 11×11 的滤波器，并绘制最后生成的滤波器的频率响应图。

```
Hd = zeros(11,11); Hd(4:8,4:8) = 1;
[f1,f2] = freqspace(11,'meshgrid');
mesh(f1,f2,Hd), axis([-1 1 -1 1 0 1.2]), colormap(jet(64))
h = fsamp2(Hd);
figure, freqz2(h,[32 32]), axis([-1 1 -1 1 0 1.2])
```

生成图 23-12，其中图（a）和图（b）分别为所需的二维频率响应和实际的二维频率响应图形。



（a）　　（b）

图 23-12　所需二维频率响应和实际二维频率响应

23.2.4　窗口法

窗口法通过将理想激励响应与一个窗口函数相乘来创建对应的滤波器。与频率取样法相似，窗口法生成一个频率响应与所需频率响应近似的滤波器。但是，窗口法往往生成比频率取样法更好的结果。

工具箱提供了两个基于窗口的滤波器设计函数，即 fwind1 和 fwind2。fwind1 函数根据指定的一个或两个一维窗口创建的二维窗口来设计二维滤波器。fwind2 函数直接用指定的二维窗口设计二维滤波器。

fwind1 函数可以使用两种方法创建二维窗口：

- 使用与旋转相似的处理，将一个一维窗口转换为一个二维窗口；
- 通过计算两个一维窗口的外积来创建一个矩形的分离窗口。

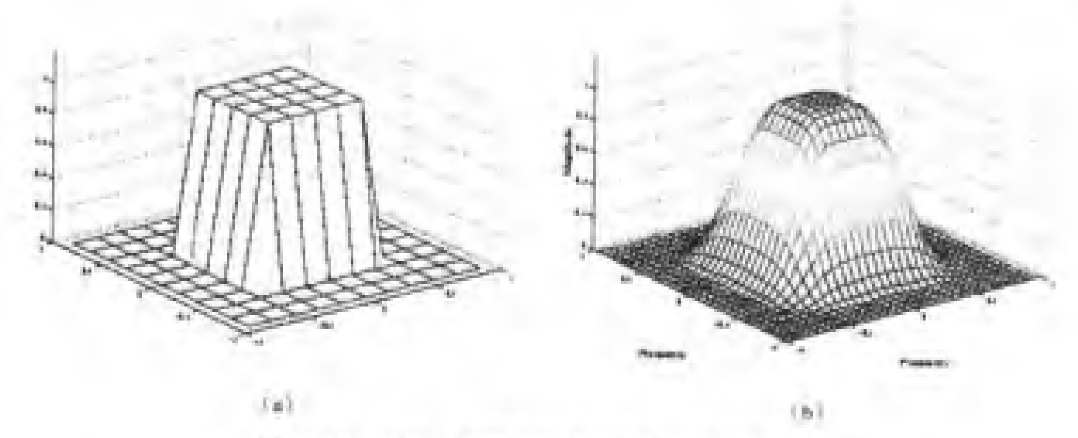
下面的例子用 fwind1 函数，利用所需的频率响应 ***Hd*** 来创建一个 11×11 的滤波器。这里，hamming 函数用于创建一个一维窗口，然后 fwind1 函数将它扩展成一个二维窗口。

```
Hd = zeros(11,11); Hd(4:8,4:8) = 1;
[f1,f2] = freqspace(11,'meshgrid');
mesh(f1,f2,Hd), axis([-1 1 -1 1 0 1.2]), colormap(jet(64))
h = fwind1(Hd,hamming(11));
figure, freqz2(h,[32 32]), axis([-1 1 -1 1 0 1.2])
```

生成图 23-13，其中图（a）和图（b）分别为所需的二维频率响应和实际的二维频率响应图。

23.2.5　创建所需频率响应矩阵

滤波器设计函数 fsamp2,fwind1 和 fwind2 都是基于所需频率响应矩阵来创建滤波器的。可以用 freqspace 函数创建一个合适的所需频率响应矩阵。该函数为任意大小的响应返回均匀间隔的频率响应值。如果用除了由 freqspace 函数返回的频率点以外的点创建所需频率响应矩阵，可能得到无法预料的结果。例如，用下面的代码行创建一个阈值为 0.5 的理想循环低通滤波频率响应。



（a）　（b）

图 23-13　所需二维频率响应和实际二维频率响应

```
[f1,f2] = freqspace(25,'meshgrid');
Hd = zeros(25,25); d = sqrt(f1.^2 + f2.^2) < 0.5;
Hd(d) = 1;
mesh(f1,f2,Hd)
```

生成图 23-14。

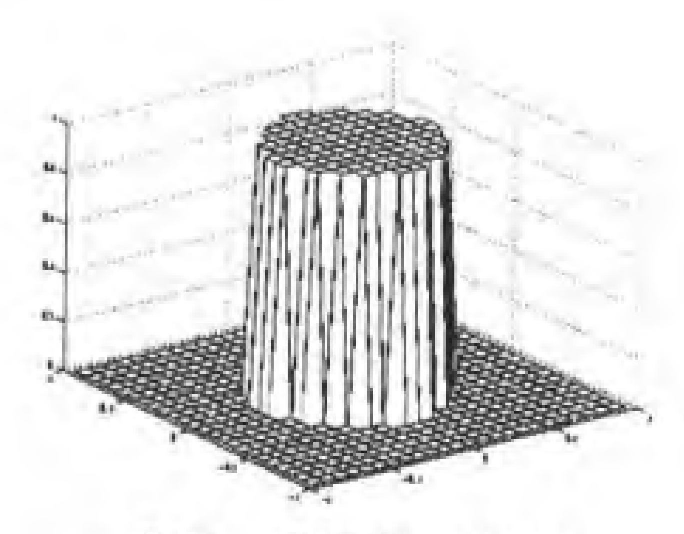


图 23-14　理想循环低通滤波频率响应

注意，对于这个频率响应，fsamp2,fwind1 和 fwind2 等函数得到的滤波器是实型的。对于大部分图像处理应用而言，这个结果是令人满意的。要想在一般情况下也达到这种效果，所需频率响应应该是关于频率原点(f1=0,f2=0)对称的。

23.2.6　计算滤波器的频率响应

freqz2 函数用二维滤波器计算频率响应。该函数没有输出变量，它生成一个频率响应的网格图。如对于下面的 FIR 滤波器：

```
h =[0.1667    0.6667    0.1667
    0.6667   -3.3333    0.6667
    0.1667    0.6667    0.1667];
```

下面的命令计算和显示 **h** 的 64×64 点频率响应。

```
freqz2(h)
```

生成图 23-15。

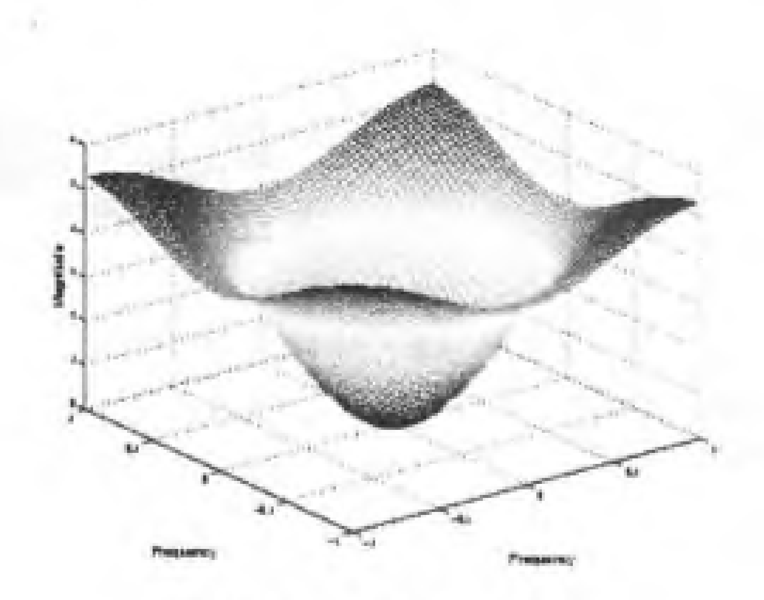


图 23-15　二维滤波器的频率响应

使用输出变量获取频率响应矩阵 **H** 和频率点向量 $f1$ 和 $f2$。

```
[H,f1,f2] = freqz2(h);
```

freqz2 函数正态化频率 f1 和 f2，这样，值 1.0 对应于取样频率的一半，或者 π 弧度。对于简单的 $m×n$ 响应，就像上面显示的那样，freqz2 函数使用了二维快速傅里叶变换函数 fft2。也可以指定任意频率点向量，但此时 freqz2 函数使用更慢的算法。

第 24 章　基于区域的处理

24.1　指定目标区域

目标区域指的是图像中要滤波或进行其他操作的部分。通过二值掩码来创建目标区域。二值掩码是一幅与要处理的图像大小相同的二值图像，其中，处于目标区域内的所有像素值为 1，目标区域外的所有像素值为 0。

下面介绍创建二值掩码的方法，包括选择一个多边形和其他选择方法。

24.1.1　选择多边形

可以用 roipoly 函数指定一个多边形目标区域。如果调用不带参数的 roipoly 函数，则鼠标光标落在图像上方时变成十字形。用鼠标在图像上单击，可以指定多边形的顶点坐标向量。选择完成以后，单击回车键，roipoly 函数返回一个与输入图像大小相同的二值图像，该图像中，落在指定多边形内部的像素值为 1，否则为 0。

下面的例子用 roipoly 函数的交互语法创建一个二值掩码。

```
I = imread('pout.tif');
imshow(I)
BW = roipoly;
```

生成图 24-1。图中，用鼠标选择的多边形边界用红色显示。



图 24-1　用 roipoly 函数选择目标多边形区域



图 24-2　创建二值掩码

```
imshow(BW)
```

生成图 24-2。

24.1.2　其他选择方法

roipoly 函数提供了一个创建二值掩码的简单方法。但是，如果二值图像与滤波图像大小相同，可以使用任何二值图像作为掩码。

例如，假设要对图像 I 滤波，而且只对那些值大于 0.5 的像素进行滤波，可以用下面的命令创建合适的掩码。

```
BW = (I > 0.5);
```

也可以用 poly2mask 函数创建二值掩码。与 roipoly 函数不同，该函数不需要输入图像。也可以用 roicolor 函数基于颜色或灰度范围来定义目标区域。这两个函数的更多内容，请参见帮助文档。

24.2　对区域进行滤波

可以用 roifilt2 函数处理目标图像。调用 roifilt2 函数时，需要指定一幅灰度图像、一个二值掩码和一个滤波器。roifilt2 函数对输入图像进行滤波并返回由经过滤波的像素值组成的图像。这种操作称为掩码滤波。

下面的例子使用掩码滤波增加图像中指定区域的对比度。

（1）读入图像。

```
I = imread('pout.tif');
```

（2）创建掩码。本例使用“选择多边形”一小节中创建的掩码 BW。掩码指定的目标区域是图中女孩夹克上的标识。

（3）创建滤波器。

```
h = fspecial('unsharp');
```

（4）调用 roifilt2 函数，指定要滤波的图像、掩码和滤波器。

```
I2 = roifilt2(h,I,BW);
imshow(I)
figure, imshow(I2)
```

生成图 24-3，其中图（a）和图（b）分别为滤波前后的图像。

roifilt2 函数还允许指定自己的函数来进行目标区域的操作。下面的例子使用 imadjust 函数来加亮图像中的部分区域。

（1）读入图像。

```
I = imread('cameraman.tif');
```

（2）创建掩码。本例中，掩码是包含文本的二值图像。掩码图像必须裁剪成与滤波后图像的大小相同。

```
BW = imread('text.png');
mask = BW(1:256,1:256);
```

（3）创建滤波器。

```
f = inline('imadjust(x,[],[],0.3)');
```


（a）


（b）

图 24-3　滤波前后的图像

（4）调用 roifilt2 函数，指定要滤波的图像、掩码和滤波器。生成的图像 I2 中含有压印的文本。

```
I2 = roifilt2(I,mask,f);
imshow(I2)
```

生成图 24-4。



图 24-4　用包含文本的二值掩码加亮图像

24.3　填充区域

可以用 roifill 函数填充目标区域，填充从目标区域的边界开始。该函数可以用于图像编辑，包括删除冗余的细节或人为污点。

roifill 函数使用基于拉普拉斯方程的插值方法进行填充操作。给定区域边界上的值时，该方法的平滑效果最佳。使用 roipoly 函数时，用鼠标选择目标区域。选择以后，roifill 函数返回一幅图像，图像中的选定区域已经被填充了。

下面的例子用 roifill 函数修改 trees 图像。选定图右侧中间位置上的一块树叶状阴影，作为要填充的区域。

```
load trees
I = ind2gray(X,map);
imshow(I)
I2 = roifill;
```

生成图 24-5。



图 24-5　选定要填充的区域

```
imshow(I2)
```

显示填充后的效果，如图 24-6 所示。



图 24-6　填充后的图形效果

第 25 章　变换域处理

25.1　傅里叶变换

傅里叶变换在图像增强、图像分析、图像恢复和图像压缩等方面扮演着重要的角色。本节介绍以下内容：

- 傅里叶变换的定义；
- 离散傅里叶变换，包括快速傅里叶变换的讨论；
- 傅里叶变换的应用。

25.1.1　傅里叶变换的定义

如果 $f(m,n)$ 是两个离散的空间变量 m 和 n 的函数，则 $f(m,n)$ 的二维傅里叶变换由以下关系式定义

$$F(\omega_1,\omega_2)=\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}f(m,n)e^{-j\omega_1 m}e^{-j\omega_2 n}$$

式中，ω_1 和 ω_2 为频率变量，单位为弧度/次。$F(\omega_1,\omega_2)$ 常被称为频域，代表 $f(m,n)$。$F(\omega_1,\omega_2)$ 是一个周期性的复数函数，周期为 2π。因为存在周期性，通常只在 $-\pi\leqslant\omega_1,\omega_2\leqslant\pi$ 范围内进行显示。注意，$F(0,0)$ 是 $f(m,n)$ 的所有值的和。所以，$F(0,0)$ 常称为傅里叶变换中的常数项或 DC 项（DC 指直流电，是电力工程学术语，表示常量电源，这里是借用）。

二维傅里叶变换的逆变换为

$$f(m,n)=\frac{1}{4\pi^2}\int_{\omega_1=-\pi}^{\pi}\int_{\omega_2=-\pi}^{\pi}F(\omega_1,\omega_2)e^{j\omega_1 m}e^{j\omega_2 n}d\omega_1\omega_2$$

图 25-1 中，对于函数 $f(m,n)$，当 m 和 n 的取值落在矩形内部时，函数值等于 1，否则等于 0。为了简化图形，$f(m,n)$ 显示为一个连续的函数，即使在 m 和 n 为离散的情况下也是如此。

图形中心的峰值为 $F(0,0)$，它是 $f(m,n)$ 中所有值的和。该图还显示高水平频率上的能量比高垂直频率上的更高。这说明 $f(m,n)$ 的水平断面上为窄脉冲，而垂直断面上为宽脉冲。

傅里叶变换的另一个可视方法是把 $\log|F(\omega_1,\omega_2)|$ 显示为一幅图像，如图 25-2 所示。

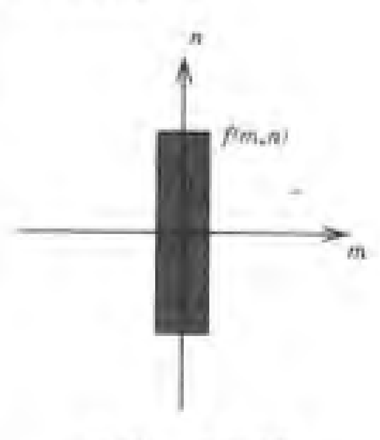


图 25-1　矩形函数

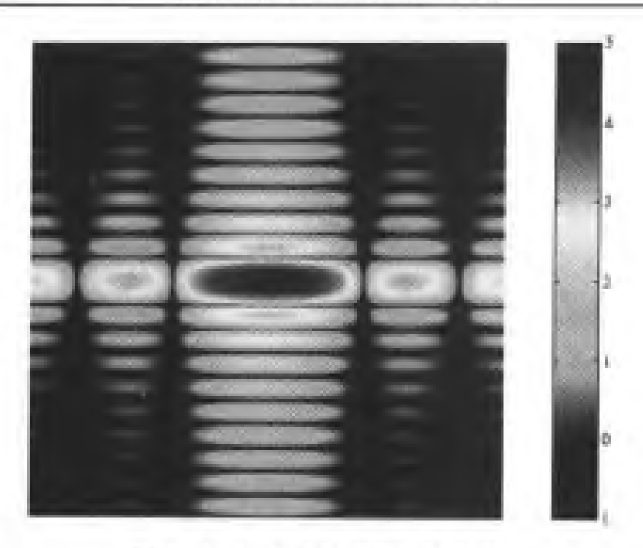


图 25-2　矩形函数傅里叶变换的对数

使用对数有助于找出 $F(\omega_1,\omega_2)$接近 0 时傅里叶变换的细节内容。图 25-3 列出了一些其他简单图形的傅里叶变换结果。

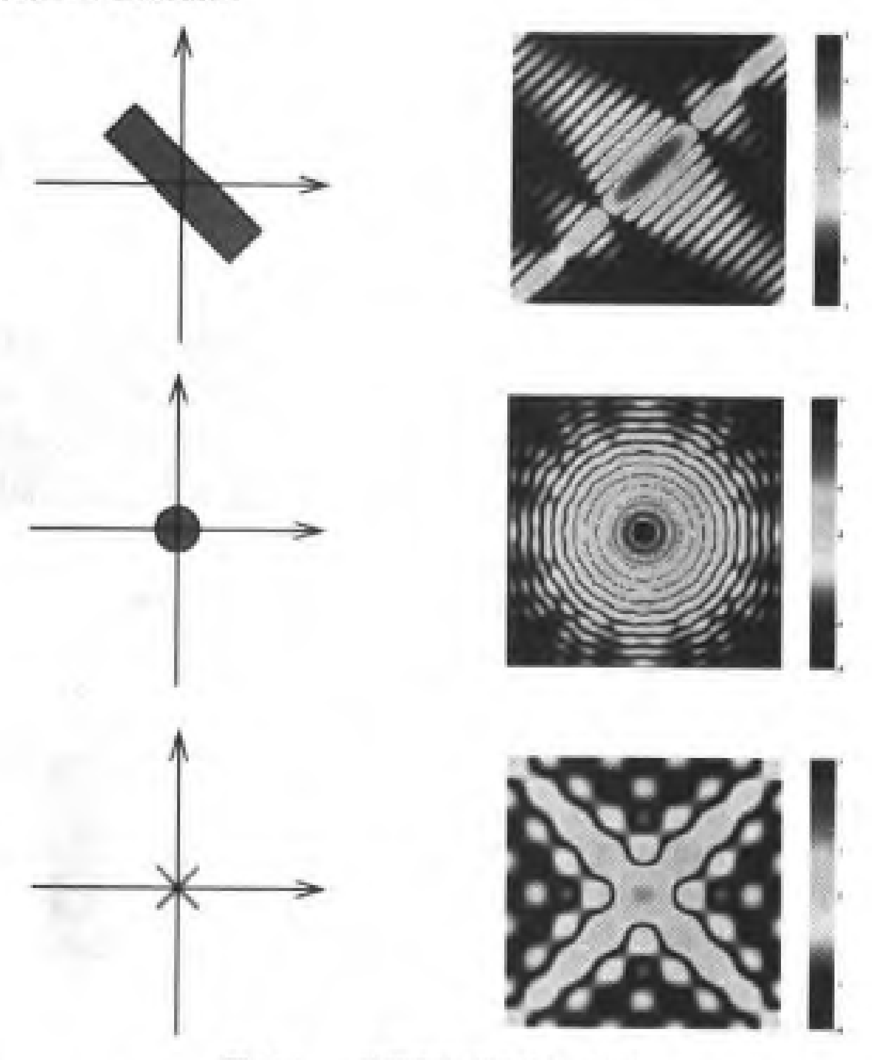


图 25-3　一些简单图形的傅里叶变换

25.1.2　离散傅里叶变换

在计算机上使用傅里叶变换常常涉及到该变换的另一种形式——离散傅里叶变换(DFT)。使用这种形式的傅里叶变换主要有以下两方面的理由：

- DFT 的输入和输出都是离散的，这使得计算机处理更加方便；
- 求解 DFT 问题有快速算法，即快速傅里叶变换(FFT)。

DFT 通常定义为一个离散的函数 $f(m,n)$，它只在有限区域 $0\leqslant m\leqslant M-1$ 和 $\pi\leqslant n\leqslant N-1$ 内是非 0 的。二维的 $M\times N$ 的 DFT 和逆 DFT 之间的关系由下式给定。

$$F(p,q)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m,n)\mathrm{e}^{-j(2\pi/M)pm}\mathrm{e}^{-j(2\pi/N)qn}\qquad \begin{array}{l}p=0,1,\ldots,M-1\\ q=0,1,\ldots,N-1\end{array}$$

式中，

$$f(m,n)=\frac{1}{MN}\sum_{p=0}^{M-1}\sum_{q=0}^{N-1}F(p,q)\mathrm{e}^{j(2\pi/M)pm}\mathrm{e}^{j(2\pi/N)qn}\qquad \begin{array}{l}m=0,1,\ldots,M-1\\ n=0,1,\ldots,N-1\end{array}$$

值 $F(p,q)$是 $f(m,n)$的 DFT 系数。MATLAB 函数 fft,fft2 和 fftn 实现了傅里叶变换算法，分别计算 1 维 DFT、2 维 DFT 和 N 维 DFT。函数 ifft, ifft2 和 ifftn 计算逆 DFT。

DFT 系数 $F(p,q)$是傅里叶变换 $F(\omega_1,\omega_2)$的特例，即

$$F(p,q)=F(\omega_1,\omega_2)\Big|_{\substack{\omega_1=2\pi p/M\\ \omega_2=2\pi q/N}}\qquad \begin{array}{l}p=0,1,\ldots,M-1\\ q=0,1,\ldots,N-1\end{array}$$

下面结合一个例子进行演示。

（1）创建一个矩阵 F，它类似于前面谈到的函数 $f(m,n)$，当 m,n 落在矩形区域内部时，函数值等于 1，否则等于 0。下面用二值图像表示 $f(m,n)$。

```
f = zeros(30,30);
f(5:24,13:17) = 1;
imshow(f,'notruesize')
```

结果如图 25-4 所示。



图 25-4　$f(m,n)$的二值图像

（2）用以下命令行计算和可视化 F 的这些大小为 30×30 的 DFT。

```
F = fft2(f);
F2 = log(abs(F));
imshow(F2,[-1 5],'notruesize'); colormap(jet); colorbar
```

生成图 25-5。

（3）为了获取傅里叶变换的更佳的取样数据，计算 ***F*** 的 DFT 时给它进行 0 填充。0 填充和 DFT 计算可以用下面的命令一步完成。

```
F = fft2(f,256,256);
```

下面的命令在计算 DFT 之前将 ***F*** 的大小 0 填充为 256×256。

```
imshow(log(abs(F)),[-1 5]); colormap(jet); colorbar
```

结果如图 25-6 所示。

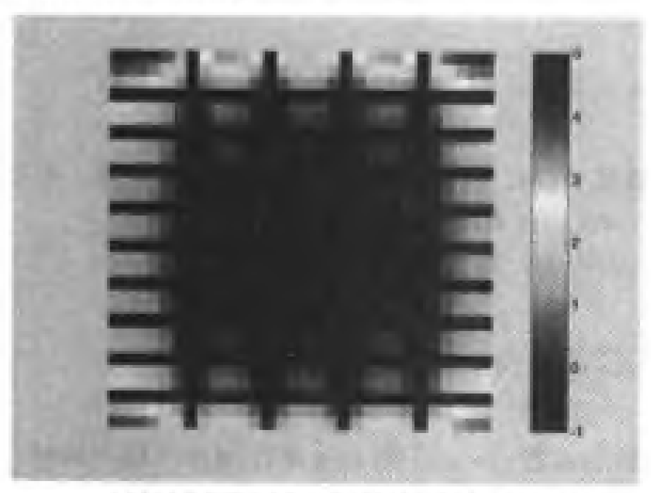


图 25-5　没有 0 填充的离散傅里叶变换

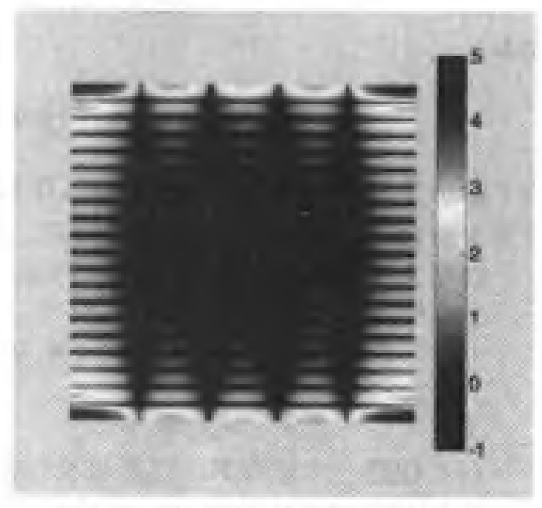


图 25-6　有 0 填充的离散傅里叶变换

（4）但是，0 频率系数仍然显示在左上角而不是中心位置。可以用 fftshift 函数解决这个问题，该函数交换 ***F*** 的象限，使得 0 频率系数位于中心位置上。

```
F = fft2(f,256,256);
F2 = fftshift(F);
imshow(log(abs(F2)),[-1 5]); colormap(jet); colorbar
```

生成图 25-7。

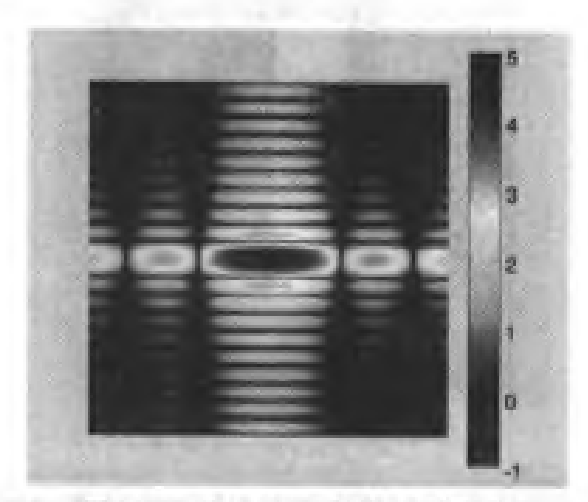


图 25-7　用 fftshift 函数处理后的图像

25.1.3 傅里叶变换的应用

下面介绍图像处理方面傅里叶变换的几种应用。

1. 线性滤波器的频率响应

线性滤波器激励响应的傅里叶变换给出了滤波器的频率响应。函数 freqz2 计算和显示滤波器的频率响应。高斯卷积核的频率响应显示该滤波器传递低频而衰减高频。

```
h = fspecial('gaussian');
freqz2(h)
```

高斯滤波器的频率响应图形如图 25-8 所示。

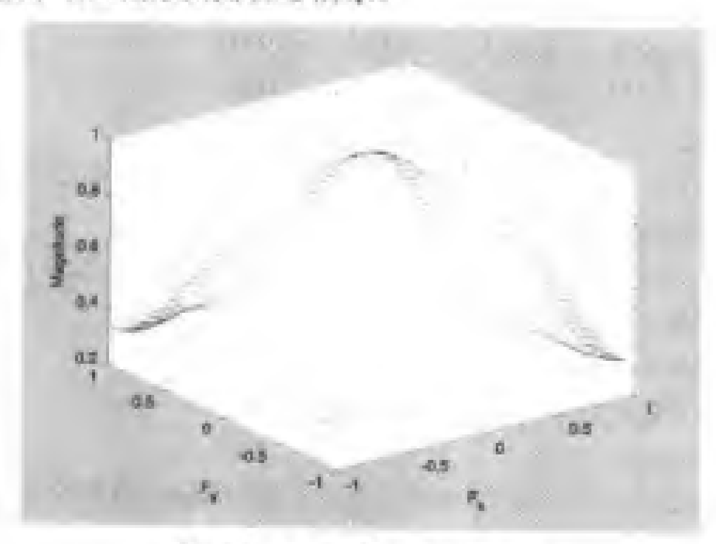


图 25-8 高斯滤波器的频率响应

1. 快速卷积

傅里叶变换的一个关键特性是，两个傅里叶变换的乘积对应于相关空间函数的卷积。这个特性与快速傅里叶变换一起，组成了快速卷积算法的基础。

注意，基于 FFT 的卷积算法主要用于输入数据量比较大的情况，对于小输入的情况，使用 imfilter 函数速度还更快。

为了进行演示，下面的例子计算 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的卷积，其中，$\boldsymbol{A}$ 是 $M\times N$ 的矩阵，$\boldsymbol{B}$ 是 $P\times Q$ 的矩阵。

（1）创建两个矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$。

```
A = magic(3);
B = ones(3);
```

（2）0 填充 $\boldsymbol{A}$ 和 $\boldsymbol{B}$，所以它们的大小至少是$(M+P-1)\times(N+Q-1)$的。本例将矩阵大小填充为 8×8。

```
A(8,8) = 0;
B(8,8) = 0;
```

（3）用 fft2 函数计算 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 的二维 DFT。

（4）将两个 DFT 相乘。

（5）用 ifft2 函数计算第 4 步结果的逆二维 DFT。

下面的代码完成第 3 至 5 步的计算。

```
C = ifft2(fft2(A).*fft2(B));
```

（6）提取结果的非 0 部分并删除圆整错误引起的虚部。

```
C = C(1:5,1:5);
C = real(C)
C =
     8.0000     9.0000    15.0000     7.0000     6.0000
    11.0000    17.0000    30.0000    19.0000    13.0000
    15.0000    30.0000    45.0000    30.0000    15.0000
     7.0000    21.0000    30.0000    23.0000     9.0000
     4.0000    13.0000    15.0000    11.0000     2.0000
```

2．查找图像特征

傅里叶变换还可以用于相关计算。相关与卷积密切相关，可以用于查找图像特征。在这里，常将相关称为模板匹配。

下面的例子演示了如何用相关来找到包含文本的图像中字母“a”出现的位置。

（1）读入示例图像。

```
bw = imread('text.png');
```

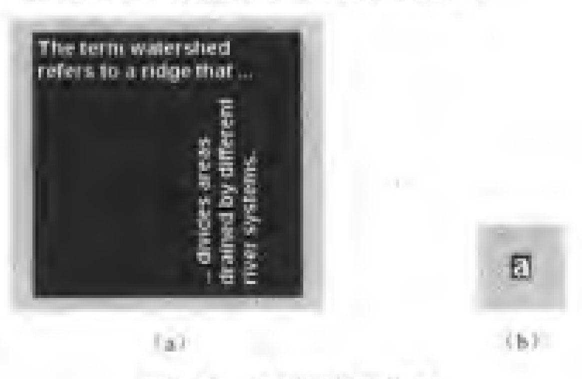
（2）通过从图像中提取出字母“a”来创建匹配模板。

```
a = bw(32:45,88:98);
```

用 pixval 函数确定图像中特征的坐标，还可以用 imcrop 函数的交互样式来创建模板图像。图中显示了原始图像和模板。

```
imshow(bw);
figure, imshow(a);
```

生成图 25-9，其中图（a）和图（b）分别为原始图像和模板。



（a）　　（b）

图 25-9　原始图像和模板

（3）将模板图像旋转 180°，然后用基于 FFT 的卷积技术计算模板图像 a 和原始图像 bw 的相关。为了使模板与图像匹配，使用 fft2 和 ifft2 函数。

```
C = real(ifft2(fft2(bw) .* fft2(rot90(a,2),256,256)));
```

图 25-10 是进行相关操作以后得到的图像。图像中的亮点对应于字母出现的位置。

```
figure, imshow(C,[])
```

生成图 25-10。

（4）为了查看图像中模板的位置，首先找到最大的像素值，然后定义一个小于这个最大值的阈值。

```
max(C(:))
ans =
    68.0000
thresh = 60;  % 将一个略小于最大值的值作为阈值
figure, imshow(C > thresh)  % 显示值超过阈值的像素
```

结果如图 25-11 所示。图中，峰值用白色小点表示。



图 25-10　相关图像



图 25-11　显示模板位置的相关阈值图像

25.2　离散余弦变换

离散余弦变换（DCT）将图像表示为具有不同振幅和频率的正弦曲线的和。图像处理工具箱中的 dct2 函数计算图像的二维离散余弦变换。DCT 的特点是，对于一幅典型的图像，图像的大部分特征可视信息可以用少数几个 DCT 系数来表征。所以，DCT 常常用于图像压缩。

大小为 $M\times N$ 的矩阵 $\boldsymbol{A}$ 的二维 DCT 可作如下定义

$$B_{pq}=\alpha_p\alpha_q\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}A_{mn}\cos\frac{\pi(2m+1)p}{2M}\cos\frac{\pi(2n+1)q}{2N},\quad \begin{array}{l}0\le p\le M-1\\0\le q\le N-1\end{array}$$

$$\alpha_p=\begin{cases}1/\sqrt{M}, & p=0\\ \sqrt{2/M}, & 1\le p\le M-1\end{cases}\qquad \alpha_q=\begin{cases}1/\sqrt{N}, & q=0\\ \sqrt{2/N}, & 1\le q\le N-1\end{cases}$$

值 $\boldsymbol{B}_{pq}$ 称为 $\boldsymbol{A}$ 的 DCT 系数。注意，MATLAB 中矩阵脚标总是从 1 而不是从 0 开始，所以，MATLAB 矩阵元素 $\boldsymbol{A}(1,1)$和 $\boldsymbol{B}(1,1)$分别对应于 A_{00} 和 B_{00}。

DCT 是可逆变换，它的逆由下式给定。

$$A_{mn}=\sum_{p=0}^{M-1}\sum_{q=0}^{N-1}\alpha_p\alpha_q B_{pq}\cos\frac{\pi(2m+1)p}{2M}\cos\frac{\pi(2n+1)q}{2N},\quad \begin{array}{l}0\le m\le M-1\\0\le n\le N-1\end{array}$$

$$\alpha_p=\begin{cases}1/\sqrt{M}, & p=0\\ \sqrt{2/M}, & 1\le p\le M-1\end{cases}\qquad \alpha_q=\begin{cases}1/\sqrt{N}, & q=0\\ \sqrt{2/N}, & 1\le q\le N-1\end{cases}$$

逆 DCT 方程可以解释为任何 $M\times N$ 矩阵 A 可以写成形式如下的 $M\times N$ 个函数的和。

$$\alpha_p\alpha_q\cos\frac{\pi(2m+1)p}{2M}\cos\frac{\pi(2n+1)q}{2N},\qquad \begin{array}{l}0\le p\le M-1\\ 0\le q\le N-1\end{array}$$

这些函数称为 DCT 的基函数。然后，DCT 系数 B_{pq} 可以看作应用于每个基函数的权重。对于 8×8 的矩阵，它的 64 个基函数可以用图 25-12 表示。

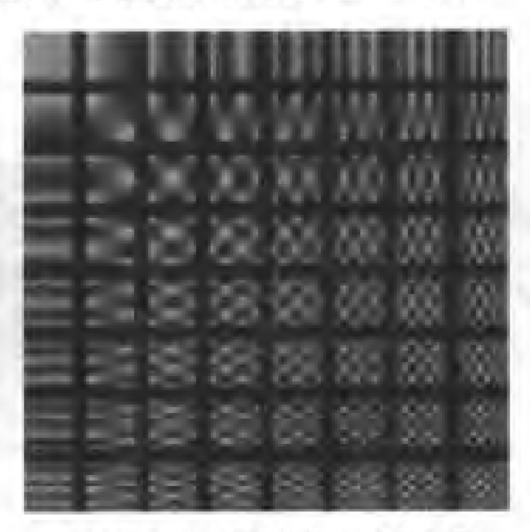


图 25-12　8×8 矩阵的 64 个基函数

水平频率从左向右增加，垂直频率从上往下增加。左上角的常值基函数通常称为 DC 基函数，对应的 DCT 系数 B_{00} 常称为 DC 系数。

25.2.1　DCT 变换矩阵

图像处理工具箱提供了两个不同的方法计算 DCT。第一个方法是使用 dct2 函数，该函数使用基于 FFT 的算法加速大输入条件下的计算。第二个方法是使用 DCT 变换矩阵，该矩阵由 dctmtx 函数返回，对于 8×8 或 16×16 的小输入情况，本方法比较有效。$M\times M$ 变换矩阵 T 如下给定。

$$T_{pq}=\begin{cases}\dfrac{1}{\sqrt{M}} & p=0,0\le q\le M-1\\ \sqrt{\dfrac{2}{M}}\cos\dfrac{\pi(2q+1)p}{2M} & 1\le p\le M-1,0\le q\le M-1\end{cases}$$

对于 $M\times M$ 的矩阵 **A**，**T*A** 是一个 $M\times M$ 的矩阵，该矩阵的列包含 **A** 各列的一维 DCT。**A** 的二维 DCT 可以用公式 **B=T*A*T**′ 计算得到。因为 **T** 是实型正交矩阵，它的逆与它的转置相同。所以，**B** 的逆二维 DCT 为 **T**′ ***B*T**。

25.2.2　DCT 和图像压缩

在 JPEG 图像压缩算法中，输入图像分割成了 8×8 或 16×16 的块，对每个块计算二维

DCT。DCT 系数然后被量子化、编码和传输。JPEG 接收器（或 JPEG 文件阅读器）解码这些量子化后的 DCT 系数，计算每个块的逆二维 DCT，然后将这些块放回到单个图像中。对于典型的图像，许多 DCT 系数的值接近于 0，丢弃它们并不严重影响重建图像的质量。

下面的代码计算输入图像中 8×8 块的二维 DCT，将每个块内 64 个 DCT 系数中的 54 个设置为 0，然后用每个块的二维逆 DCT 重建图像。这里使用了变换矩阵计算方法。

```
I = imread('cameraman.tif');
I = im2double(I);
T = dctmtx(8);
B = blkproc(I,[8 8],'P1*x*P2',T,T');
mask = [1   1   1   1   0   0   0   0
        1   1   1   0   0   0   0   0
        1   1   0   0   0   0   0   0
        1   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0
        0   0   0   0   0   0   0   0];
B2 = blkproc(B,[8 8],'P1.*x',mask);
I2 = blkproc(B2,[8 8],'P1*x*P2',T',T);
imshow(I), figure, imshow(I2)
```

重建前后的图像如图 25-13 所示。



(a)



(b)

图 25-13　重建前后的图像

尽管重建后的图像有些质量损失，但是图像中的对象仍然清晰可辨，即使在丢弃 85% 的 DCT 系数的情况下也是如此。

25.3　Radon 变换

25.3.1　概念

图像处理工具箱中的 radon 函数计算指定方向上图像矩阵的投影。二维函数 $f(x,y)$的投

影是一组线积分。radon 函数计算一定方向上平行光束的线积分，光线间隔 1 个像素单位。为了表示图像，radon 函数通过围绕图像中心旋转光源来从不同角度获得图像的平行光投影。

可以沿任意角度θ计算投影。通常，$f(x,y)$的 Radon 变换是平行于 y 轴的f的线积分。

$$R_\theta(x') = \int_{-\infty}^{\infty} f(x'\cos\theta - y'\sin\theta, x'\sin\theta + y'\cos\theta)dy'$$

式中，

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

图 25-14 显示了 Radon 变换的几何表示。

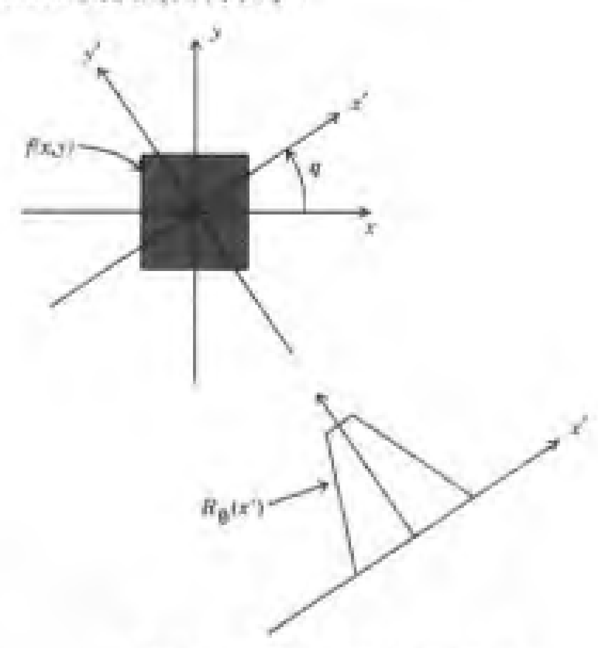


图 25-14　Radon 变换的几何表示

下面的命令行计算图像 I 的 Radon 变换，旋转角度在 theta 参数中指定。

```
[R,xp] = radon(I,theta);
```

R 的列包含了 ***theta*** 中每个角度的 Radon 变换。向量 ***xp*** 包含沿 x' 轴的对应坐标。I 的中心像素定义为 floor((size(I)+1)/2)，它是 x' 轴上对应于 $x'=0$ 的像素。

下面的命令行计算和显示一幅包含一个方形对象的图像在 0 度和 45 度上的 Radon 变换。

```
I = zeros(100,100);
I(25:75, 25:75) = 1;
imshow(I)
```

图像如图 25-15 所示。

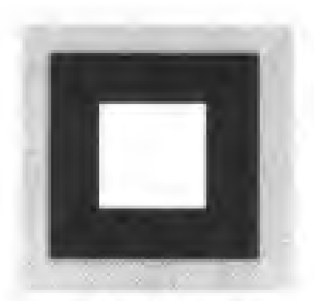
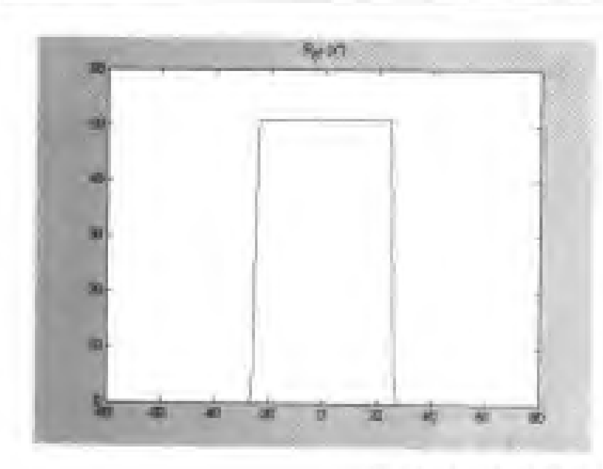


图 25-15　图像

用下面的命令行进行变换：

```
[R,xp] = radon(I,[0 45]);
figure; plot(xp,R(:,1)); title('R_{0^o} (x\prime)')
figure; plot(xp,R(:,2)); title('R_{45^o} (x\prime)')
```

变换结果如图 25-16 所示。



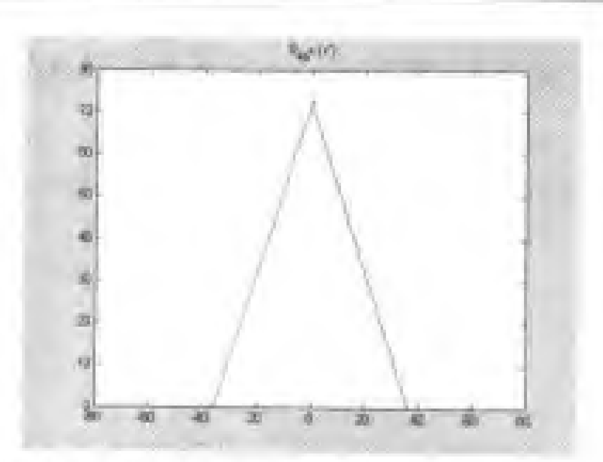


图 25-16　方形图像的两个 Radon 变换结果

角度很多时，Radon 变换的结果常常用图像显示。下面的例子中，方形图像的 Radon 变换从 0 度一直计算到 180 度，间隔为 1 度。

```
theta = 0:180;
[R,xp] = radon(I,theta);
imagesc(theta,xp,R);
title('R_{\theta} (X\prime)');
xlabel('\theta (degrees)');
ylabel('X\prime');
set(gca,'XTick',0:20:180);
colormap(hot);
colorbar
```

结果如图 25-17 所示。

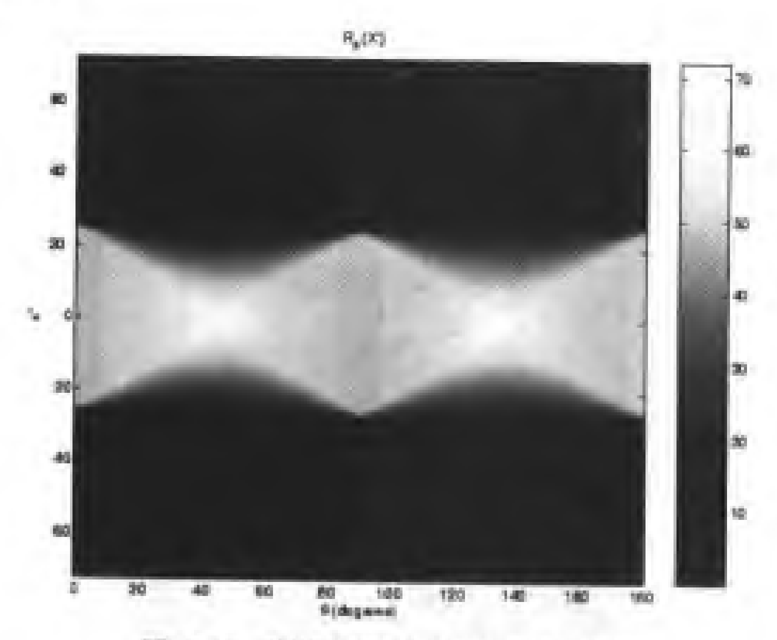


图 25-17　使用了 180 次投影的 Radon 变换

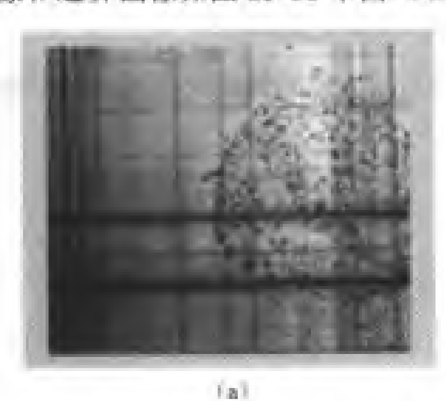
25.3.2 使用 Radon 变换来发现线形影像

Radon 变换与 Hough 变换紧密相关。可以用 radon 函数实现 Hough 变换的一种形式来找出图像中的直线形对象。按照如下步骤进行：

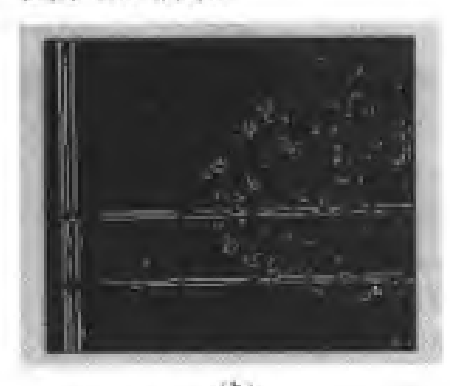
（1）用 edge 函数计算二值边界图像。

```
I = fitsread('solarspectra.fts');
I = mat2gray(I);
BW = edge(I);
imshow(I), figure, imshow(BW)
```

原始图像和边界图像如图 25-18 中图（a）和图（b）所示。



（a）



（b）

图 25-18　用 Radon 变换发现线形影象

（2）计算边界图像的 Radon 变换。

```
theta = 0:179;
[R,xp] = radon(BW,theta);
figure, imagesc(theta, xp, R); colormap(hot);
xlabel('\theta (degrees)'); ylabel('X\prime');
title('R_{\theta} (X\prime)');
colorbar
```

边界图像的 Radon 变换图像如图 25-19 所示。

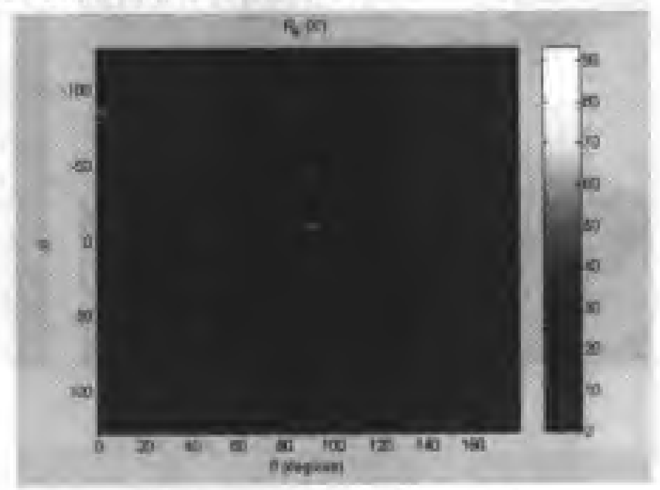


图 25-19　边界图像的 Radon 变换

25.3.3　逆 Radon 变换

用 iradon 函数进行逆 Radon 变换。该变换通常用于 X 射线断层摄影术的应用。这个变换转化 Radon 变换，所以可以根据投影数据进行图像重建。

下面首先用 radon 函数计算图像 I 在一组旋转角度 *theta* 下的 Radon 变换 ***R***，然后用 iradon 函数根据 ***R*** 和 theta 重建图像 I。

```
R = radon(I,theta);
IR = iradon(R,theta);
```

上例中，投影是从原始图像 I 计算得到的。但是在大部分应用领域，并没有形成投影的原始图像。例如，在 X 射线断层摄影应用中，投影是通过度量以不同角度穿过身体的射线的衰减情况来组成的。原始图像可以认为是人体的横断面，图像灰度表示人体的密度。投影通过特制的设备来搜集，然后用 iradon 函数根据这些投影来重建人体图像。利用这种技术，可以在不侵入人体或其他不透明对象内部的情况下获得它们的图像。

iradon 函数根据平行光投影来重建图像，每个投影由指定角度上的一组线积分组合而成。

图 25-20 显示了平行光投影在 X 射线断层摄影方面的应用。注意，这里发射器和传感器的数量是相等的。每个传感器测量对应发射器发出的射线，根据射线的衰减可以计算对象的总体密度。

图 25-20 演示了穿过对象的平行线束。$f(x,y)$表示图像的亮度，$R_\theta(x')$是角度 *theta* 上的投影。

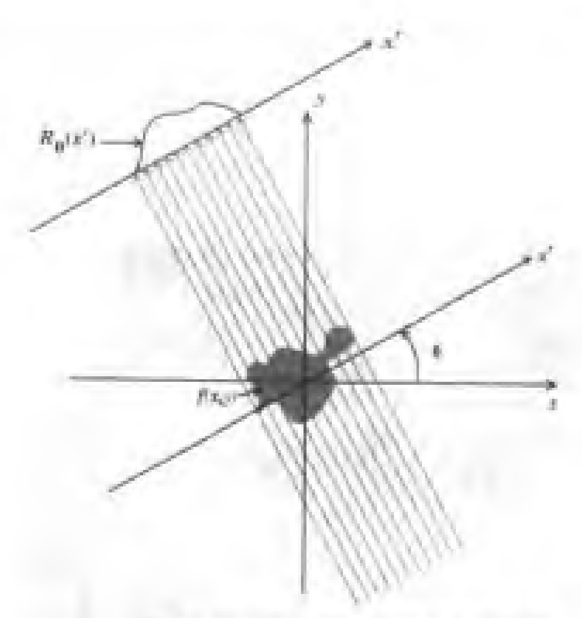


图 25-20　穿过对象的平行光束投影

25.3.4　利用投影数据重建图像

下面的代码演示了如何利用平行投影数据重建图像。测试图像是 Shepp-Logan 头部剖视图，可以用工具箱中的 phantom 函数生成。剖视图中外部的白色椭圆形壳可看作头骨，内部的许多椭圆形可看作脑部组织。

（1）生成 Shepp-Logan 头部剖视图图像。

```
P = phantom(256);
imshow(P)
```

剖视图图像如图 25-21 所示。

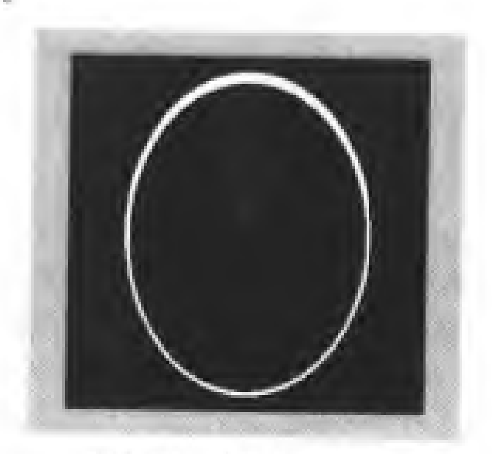


图 25-21　头部剖视图图像

（2）计算三套 *theta* 值对应剖视图的 Radon 变换。

```
theta1 = 0:10:170; [R1,xp] = radon(P,theta1);
theta2 = 0:5:175;  [R2,xp] = radon(P,theta2);
theta3 = 0:2:178;  [R3,xp] = radon(P,theta3);
```

（3）显示一副 Shepp-Logan 头部剖视图的 Radon 变换图。显示第 3 套 theta 值对应的图像，对应的变换有 90 次投影。

```
figure, imagesc(theta3,xp,R3); colormap(hot); colorbar
xlabel('\theta'); ylabel('x\prime');
```

变换结果如图 25-22 所示。

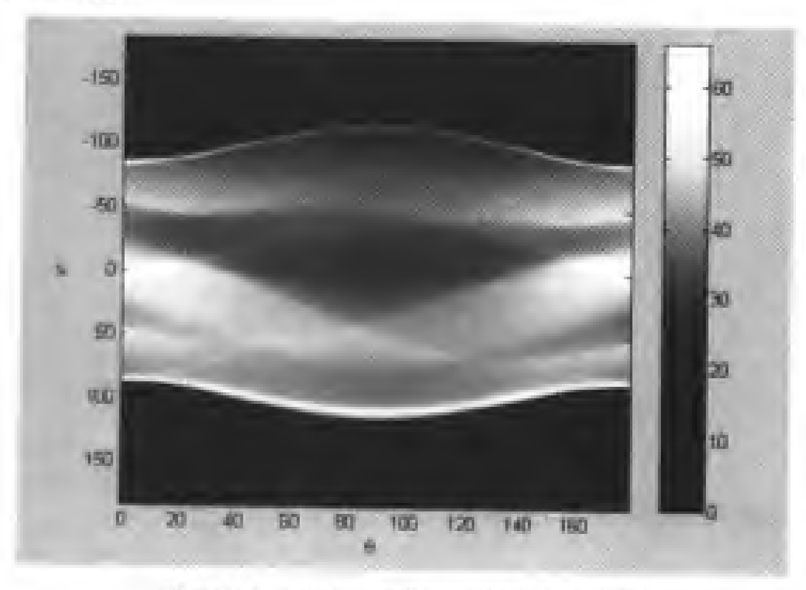
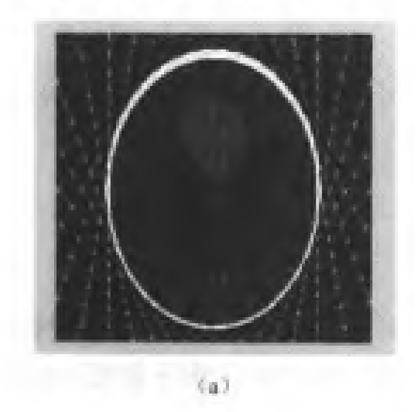


图 25-22　使用了 90 次投影的 Radon 变换

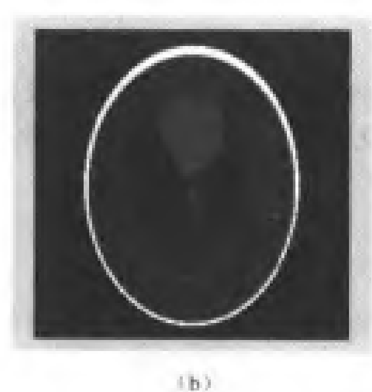
（4）根据第 2 步生成的投影数据重建头部剖视图图像。

```
I1 = iradon(R1,10);
I2 = iradon(R2,5);
I3 = iradon(R3,2);
imshow(I1)
figure, imshow(I2)
figure, imshow(I3)
```

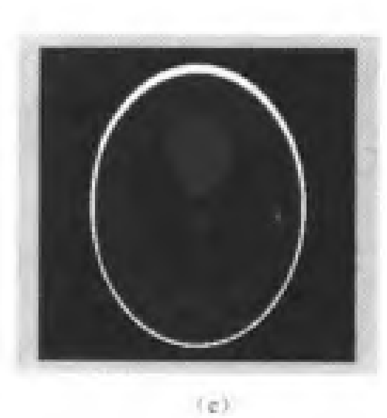
重建结果如图 25-23 所示。



(a)



(b)



(c)

图 25-23　头部影像的重建结果

第 26 章　数学形态学

形态学是基于形状的图像处理技术。输出图像中每个像素的值都是输入图像中该像素与相邻像素比较后的结果。通过选择邻域的大小和形状，可以完成对输入图像中特定形状敏感的数学形态学处理。

本章介绍图像处理工具箱中的数学形态学函数。可以使用这些函数完成常见的图形处理任务，如对比增强、滤波、细化、骨架提取、填充和分割等。

26.1　膨胀和腐蚀

膨胀和腐蚀是两个基本的数学形态学运算。膨胀是将像素添加到图像中物体的边缘，腐蚀则是删除对象边缘的像素。添加或删除的像素数目与用于处理图像的结构元素的大小和形状有关。

下面介绍的内容有：

- 提供有关膨胀和腐蚀函数如何工作的重要背景信息；
- 描述结构元素和如何创建它们；
- 描述如何进行数学形态学膨胀；
- 描述如何进行数学形态学腐蚀；
- 介绍一些基于膨胀和腐蚀的常见操作；
- 介绍基于膨胀和腐蚀的工具箱函数。

26.1.1　理解膨胀和腐蚀

在数学形态学膨胀和腐蚀运算中，输出图像中任何已知像素的状态都是通过给输入图像中的对应像素和其相邻像素采用一个运算规则来确定的。表 26-1 列出了用于膨胀和腐蚀的规则。

表 26-1　灰度膨胀和腐蚀的规则

运　算	规　　则
膨胀	输出像素的值是输入像素所有相邻像素值的最大值。二值图像中，如果任何相邻像素的值为 1，则输出像素的值设置为 1
腐蚀	输出像素的值是输入像素所有相邻像素值的最小值。二值图像中，如果任何相邻像素的值为 0，则输出像素的值设置为 0

图 26-1 演示了二值图像的膨胀。注意结构元素是如何定义目标像素（用圆圈圈起来的那个像素）的邻域的。膨胀函数对相邻像素采用了合适的规则，并且给输出图像中的对应像素指定了一个值。图中，数学形态学函数将输出像素的值设置为 1，因为结构元素所定义的邻域内有一个像素的值是 1。

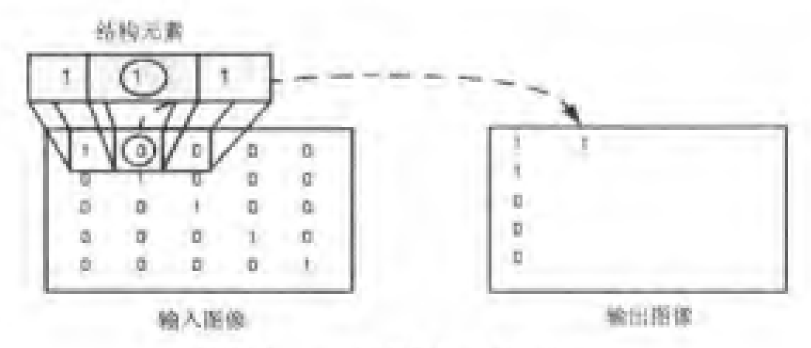


图 26-1　二值图像的数学形态学膨胀

图 26-2 演示了灰度图像的处理。该图显示了输入图像中特定像素的处理过程。注意函数是如何采用规则来确定输入像素的邻域并把邻域内所有像素的最高值作为输出图像中对应像素的值的。

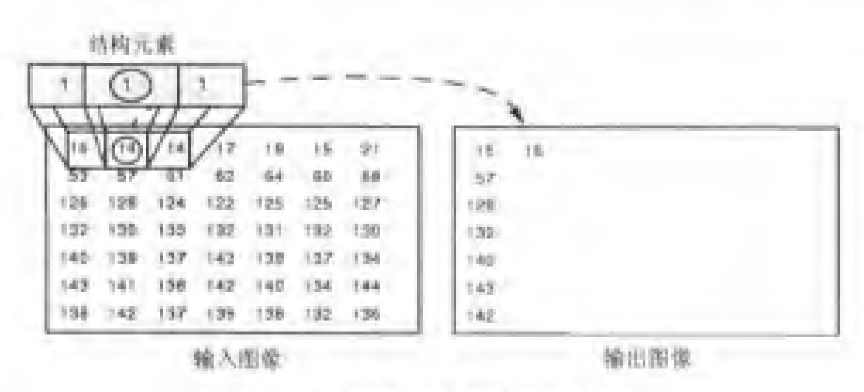


图 26-2　灰度图像的数学形态学膨胀

26.1.2　结构元素

膨胀和腐蚀运算的核心内容就是结构元素。二维，或者说平面结构元素由元素为 0 和 1 的矩阵组成，一般比进行处理的图像要小得多。结构元素的中心像素，即原点可以确定目标像素。结构元素中包含 1 的像素定义结构元素的邻域。这些像素在进行膨胀和腐蚀时都要进行考虑。

三维，或者说非平面的结构元素用 0 和 1 定义结构元素在 x-y 平面上的范围，用高度值定义第三维。

1．结构元素的原点

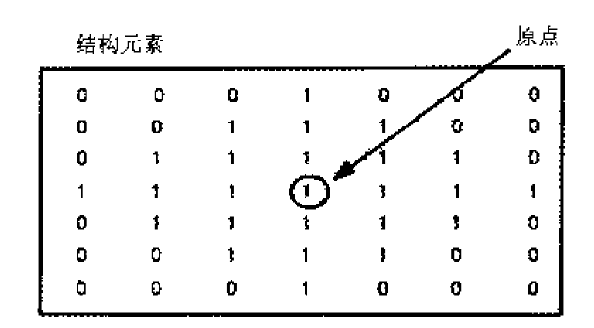
用下面的命令行获取任何大小和维数的结构元素的原点坐标。

```
origin = floor((size(nhood)+1)/2)
```

代码中，nhood 是用结构元素定义的邻域。因为结构元素是 MATLAB 对象，所以计算中不能使用 STREL 对象本身的大小。必须使用 getnhood 方法从 STREL 对象中提取结构元素的邻域。例如，图 26-3 显示了一个菱形的结构元素。

2．创建结构元素

工具箱中的膨胀和腐蚀函数接受名为 STREL 的结构元素对象。可以用 strel 函数创建任何大小和形状的 STREL。该函数还支持一些通用形状如直线、菱形、圆形和球形等。



0	0	0	1	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
1	1	1	(1)	1	1	1
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0

图 26-3　一个菱形结构元素的原点

注意，通常会选择与输入图像中要处理的对象大小和形状相同的结构元素。例如，要想在图像中查找直线段，可以创建线形结构元素。

例如，下面的代码创建一个平面的菱形结构元素。

```
se = strel('diamond',3)
se =
Flat STREL object containing 25 neighbors.
Decomposition: 3 STREL objects containing a total of 13 neighbors

Neighborhood:
     0     0     0     1     0     0     0
     0     0     1     1     1     0     0
     0     1     1     1     1     1     0
     1     1     1     1     1     1     1
     0     1     1     1     1     1     0
     0     0     1     1     1     0     0
     0     0     0     1     0     0     0
```

3．结构元素分解

为了增强效果，strel 函数可能会将结构元素分成更小的部分，可以用一种称为结构元素分解的方法来实现。

例如，用一个 11×11 的方形结构元素进行膨胀，可以首先用一个 1×11 的结构元素进行膨胀，然后用一个 11×1 的结构元素再次进行膨胀。在理论上，这样处理将会使计算提速 5.5 倍，当然，实际效果要慢一点点。

结构元素分解不能用于任意结构元素，除非它是一个邻域都由 1 组成的平面结构元素。

用 getsequence 方法查看分解中使用的结构元素序列。该函数返回一个形成分解的结构元素数组。例如，下面是菱形结构元素分解中创建的结构元素。

```
sel = strel('diamond',4)
sel =
Flat STREL object containing 41 neighbors.
Decomposition: 3 STREL objects containing a total of 13 neighbors

Neighborhood:
```

```
     0     0     0     0     1     0     0     0     0
     0     0     0     1     1     1     0     0     0
     0     0     1     1     1     1     1     0     0
     0     1     1     1     1     1     1     1     0
     1     1     1     1     1     1     1     1     1
     0     1     1     1     1     1     1     1     0
     0     0     1     1     1     1     1     0     0
     0     0     0     1     1     1     0     0     0
     0     0     0     0     1     0     0     0     0

seq = getsequence(sel)
seq =
3x1 array of STREL objects

seq(1)
ans =
Flat STREL object containing 5 neighbors.

Neighborhood:
     0     1     0
     1     1     1
     0     1     0

seq(2)
ans =
Flat STREL object containing 4 neighbors.

Neighborhood:
     0     1     0
     1     0     1
     0     1     0

seq(3)
ans =
Flat STREL object containing 4 neighbors.

Neighborhood:
     0     0     1     0     0
     0     0     0     0     0
     1     0     0     0     1
     0     0     0     0     0
     0     0     1     0     0
```

26.1.3 处理图像边缘的像素

数学形态学函数在输入图像中的目标像素上确定结构元素原点的位置。对于图像边缘上的像素，结构元素所定义的部分邻域可以扩展到图像边界以外。

为了处理边界像素，数学形态学函数给这些没有定义的像素指定了一个值，就好像这些函数已经用额外的行和列填补了图像。这些填补的像素根据膨胀和腐蚀运算有所不同。表26-2介绍了二值图像和灰度图像两种情况下的膨胀和腐蚀规则。

表26-2 填补图像的规则

运 算	规 则
膨胀	图像边界外的像素值为所提供的数据类型的最小值。对于二值图像，假定这些像素的值为0。对于灰度图像，对于uint8型图像，最小值为0
腐蚀	图像边界外的像素值为所提供的数据类型的最大值。对于二值图像，假定这些像素的值为1。对于灰度图像，对于uint8型图像，最大值为255

注意，通过使用最小值进行膨胀运算和使用最大值进行腐蚀运算，工具箱避免了边界效应问题。所谓边界效应，指的是输出图像边缘附近的区域看起来不像图像的其余部分均一。例如，如果通过填补最小值来进行腐蚀，将导致输出图像周边出现一条黑边。

26.1.4 膨胀图像

用imdilate函数膨胀图像。imdilate函数接受下面两个主要变量：

- 要处理的输入图像（灰度图像、二值图像或压缩二值图像）；
- 由strel函数返回的结构元素对象或定义结构元素邻域的二值矩阵。

imdilate函数还接受两个可选变量：PADOPT和PACKOPT。PADOPT变量影响输出图像的尺寸。PACKOPT变量把输入图像识别为压缩二值图像。

下面的例子对一幅包含矩形对象的简单二值图像进行膨胀操作。

```
BW = zeros(9,10);
BW(4:6,4:7) = 1
BW =
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     1     1     1     1     0     0     0
     0     0     0     1     1     1     1     0     0     0
     0     0     0     1     1     1     1     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```

为了扩展前景对象的各个边，本例使用了一个3×3的方形结构元素对象。

```
SE = strel('square',3)
SE =
Flat STREL object containing 3 neighbors.
Neighborhood:
     1     1     1
     1     1     1
     1     1     1
```

为了膨胀图像，给 imdilate 函数传递图像 BW 和结构元素 SE。注意膨胀操作是如何给前景对象周围添加一圈 1 的。

```
BW2 = imdilate(BW,SE)
BW2 =
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     1     1     1     1     1     1     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```

26.1.5 腐蚀图像

用 imerode 函数腐蚀图像。该函数接受两个主要的变量：

- 要处理的输入图像（灰度图像、二值图像或压缩二值图像）；
- 由 strel 函数返回的结构元素对象或定义结构元素邻域的二值矩阵。

imerode 函数还接受 3 个可选变量：PADOPT,PACKOPT 和 M。PADOPT 变量影响输出图像的大小。PACKOPT 变量把输入图像识别为压缩二值图像。如果图像是压缩二值图像，则 M 确定原始图像中的行数。

下面的例子腐蚀一幅二值图像 circbw.tif。

（1）把图像读入到 MATLAB 工作空间。

```
BW1 = imread('circbw.tif');
```

（2）创建一个结构元素。下面的代码创建一个对角形结构元素对象。

```
SE = strel('arbitrary',eye(5));
SE=
Flat STREL object containing 5 neighbors.
Neighborhood:
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

（3）调用 imerode 函数，把图像 BW 和结构元素 SE 作为变量进行传递。

```
BW2 = imerode(BW1,SE);
imshow(BW1)
figure, imshow(BW2)
```

生成图 26-4，其中图（a）和图（b）分别为腐蚀前后的图像。注意图（b）中的输出图像加上了很多对角斜条纹，这是因为结构元素的形状是这样的。



（a）



（b）

图 26-4　腐蚀前后的图像

26.1.6　组合膨胀和腐蚀

常将膨胀和腐蚀组合起来完成一定的图像处理任务。例如，图像的数学形态学开运算的实现就是用相同的结构元素先后进行腐蚀和膨胀。相关的闭运算与此相反。

下面用 imdilate 函数和 imerode 函数演示如何实现数学形态学的开运算。但是，注意，工具箱已经包含了 imopen 函数，它可以完成这个任务。

可以用数学形态学开运算的方法从图像中删除小对象，并且同时保持图像中大对象的形状和大小不变。例如，可以用 imopen 函数从原始电路图 circbw.tif 中删除所有电路线，创建一个只有微芯片矩形形状的输出图像。

按以下步骤进行操作。

（1）把图像读入 MATLAB 工作空间。

```
BW1 = imread('circbw.tif');
```

（2）创建结构元素。

```
SE = strel('rectangle',[40 30]);
```

结构元素应该大小合适，以便腐蚀图像时只能删除直线而不能删除矩形。它应该全部由 1 组成，因而可以删除除了大的连续前景像素片以外的所有对象。

（3）用上面创建的结构元素腐蚀图像。

```
BW2 = imerode(BW1,SE);
imshow(BW2)
```

腐蚀结果如图 26-5 所示。可以看出，这一步操作删除了所有直线段，但是把矩形也收缩小了。

为了把矩形恢复到原始大小，用相同的结构元素 SE 膨胀图像。

```
BW3 = imdilate(BW2,SE);
imshow(BW3)
```

膨胀结果如图 26-6 所示。



图 26-5　腐蚀结果



图 26-6　膨胀结果

26.1.7　基于膨胀和腐蚀的函数

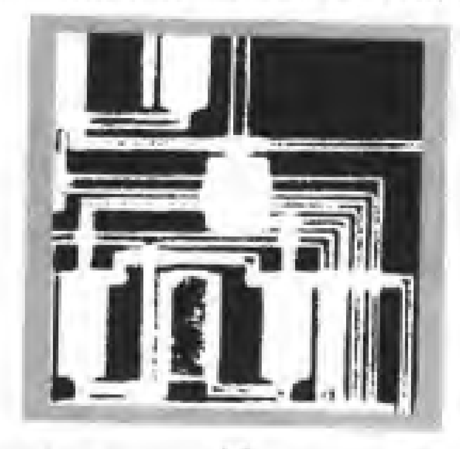
下面介绍两个基于膨胀和腐蚀的通用图像处理操作：骨架提取和边界检测。除了这两个操作外，工具箱还提供了进行图像开运算和闭运算等操作的函数，请参见帮助文档，不再一一介绍。

1．骨架提取

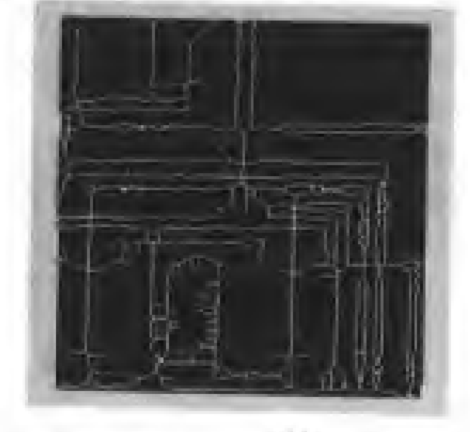
使用 bwmorph 函数，在不改变图像基本结构的情况下，可以将图像中的所有对象简化为线条表示。这个处理方法称为骨架提取。

```
BW1 = imread('circbw.tif');
BW2 = bwmorph(BW1,'skel',Inf);
imshow(BW1)
figure, imshow(BW2)
```

结果如图 26-7 所示，其中图（a）和图（b）分别为骨架提取前后的图像。



（a）



（b）

图 26-7　骨架提取前后的图像

2．边缘检测

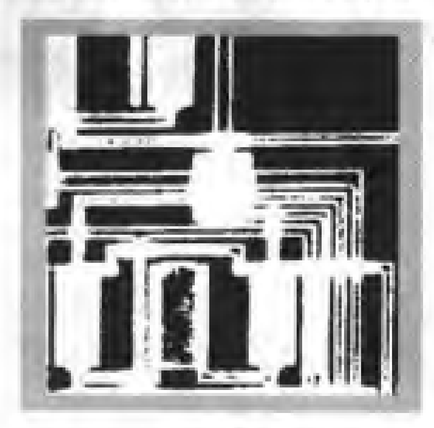
bwperim 函数确定二值图像中对象的边缘像素。如果一个像素满足下面这些条件，那么它可以看作边缘像素：

- 像素状态为 on；
- 它周围有一个以上像素的状态为 off。

例如，下面的代码查找电路板二值图像中的边缘像素。

```
BW1 = imread('circbw.tif');
BW2 = bwperim(BW1);
imshow(BW1)
figure, imshow(BW2)
```

边缘检测结果如图 26-8 所示，其中图（a）和图（b）分别为边缘检测前后的图像。



（a）



（b）

图 26-8　边缘检测前后的图像

26.2　数学形态学重建

数学形态学重建是数学形态学图像处理的另一个重要内容。基于膨胀，数学形态学重建有下面一些特性：

- 处理基于两幅图像，而不是一幅图像和一个结构元素；
- 进行重复处理直到稳定（即图像不再改变）；
- 处理基于连通性，而不是基于结构元素。

本节介绍以下内容：

- 提供数学形态学重建的背景信息，介绍如何使用 imreconstruct 函数；
- 介绍像素连通性如何影响数学形态学重建；
- 介绍如何使用 imfill 函数，它基于数学形态学重建；
- 介绍一组基于数学形态学重建的函数。

26.2.1　Marker 图像和 Mask 图像

数学形态学重建根据另一幅图像（Mask 图像）的特点处理当前图像（Marker 图像）。

Marker 图像中的高点或峰值点指示从哪里开始处理。处理会一直继续下去，直到图像的值不再发生改变。

为了演示数学形态学重建，考虑下面的简单图像。它包括两个主要的区域，即分别包含数字 14 和 18 的区块。背景主要设置为 10，有些像素设置为 11。

```
A=[10 10 10 10 10 10 10 10 10 10;
   10 14 14 14 10 10 11 10 11 10;
   10 14 14 14 10 10 10 11 10 10;
   10 14 14 14 10 10 11 10 11 10;
   10 10 10 10 10 10 10 10 10 10;
   10 11 10 10 10 18 18 18 10 10;
   10 10 10 11 10 18 18 18 10 10;
   10 10 11 10 10 18 18 18 10 10;
   10 11 10 11 10 10 10 10 10 10;
   10 10 10 10 10 10 11 10 10 10];
```

按照下面的步骤，从数学形态学上重建这幅图像。

（1）创建一幅 Marker 图像。就像膨胀和腐蚀中的结构元素一样，Marker 图像的特点决定数学形态学重建中的处理过程。Marker 图像中的峰值应该识别希望强调的 Mask 图像中对象的位置。

创建 Marker 图像的一种方法是用 imsubtract 函数从 Mask 图像中减去一个常数。

```
Marker = imsubtract(A,2)
Marker =
     8     8     8     8     8     8     8     8     8     8
     8    12    12    12     8     8     9     8     9     8
     8    12    12    12     8     8     8     9     8     8
     8    12    12    12     8     8     9     8     9     8
     8     8     8     8     8     8     8     8     8     8
     8     9     8     8     8    16    16    16     8     8
     8     8     8     9     8    16    16    16     8     8
     8     8     9     8     8    16    16    16     8     8
     8     9     8     9     8     8     8     8     8     8
     8     8     8     8     8     8     9     8     8     8
```

（2）调用 imreconstruct 函数，完成图像的数学形态学重建。在输出图像中，注意除了亮度峰值以外的所有亮度波动是如何被删除的。

```
recon = imreconstruct(Marker, A)
recon =
    10    10    10    10    10    10    10    10    10    10
    10    12    12    12    10    10    10    10    10    10
    10    12    12    12    10    10    10    10    10    10
    10    12    12    12    10    10    10    10    10    10
    10    10    10    10    10    10    10    10    10    10
    10    10    10    10    10    16    16    16    10    10
    10    10    10    10    10    16    16    16    10    10
```

```
10   10   10   10   10   16   16   16   10   10
10   10   10   10   10   10   10   10   10   10
10   10   10   10   10   10   10   10   10   10
```

数学形态学重建在概念上可以理解为 Marker 图像反复膨胀，直到 Marker 图像的轮廓与 Mask 图像的相吻合。此时，Marker 图像中的峰值会“展开”或膨胀。

图 26-9 演示了一维条件下的数学形态学重建。每个连续的膨胀必须位于 Mask 图像的下方。当进一步膨胀时如果图像停止变化，处理停止。最后一次膨胀的结果就是重建的图像。注意，工具箱中本操作的实际实现要方便得多。图中显示了 Marker 图像的连续膨胀。

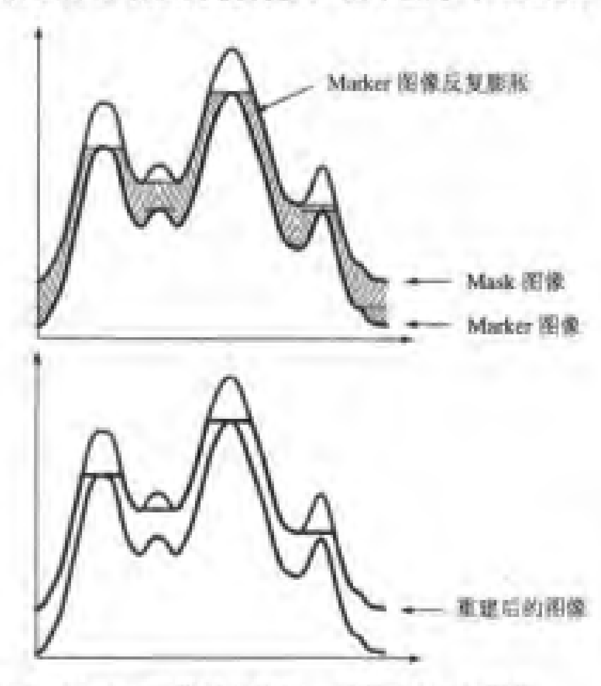


图 26-9　Marker 图像反复膨胀，受限于 Mask 图像

26.2.2　像素连通性

数学形态学处理从 Marker 图像的峰值处开始，并且基于像素连通性向图像其他部分扩展。连通性定义哪些像素与其他像素相连。

例如，下面的二值图像包含一个前景对象——它的所有像素值都设置为 1。如果背景是 4 连通的，则图像有一个背景对象，并且所有像素的值都设置为 0。但是，如果前景是 8 连通的，则前景形成一个闭合回路，并且图像有两个单独的背景：回路内部的背景和回路外部的背景。

```
0   0   0   0   0   0   0   0
0   1   1   1   1   1   0   0
0   1   0   0   0   1   0   0
0   1   0   0   0   1   0   0
0   1   0   0   0   1   0   0
0   1   1   1   1   0   0   0
0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0
```

表 26-3 和表 26-4 分别列出了工具箱支持的所有标准二维和三维连通性。

表 26-3　二维连通性

4 连通	共边的像素连通。这表示只有在相连像素对都在水平或垂直方向上，并且相连的情况下才会成为同一对象的一部分	
8 连通	共边和共点的像素都连通。表示不管两个像素是否在水平、垂直或对角方向上连通，只要它们都在这个方向上，它们就是同一对象的一部分	

表 26-4　三维连通性

6 连通	如果共面，则像素连通	
18 连通	如果共面或共边，则像素连通	
26 连通	如果像素共面、共边或共顶点，则像素连通	

邻域的类型会影响图像中找到的对象个数和这些对象的边界的条数。所以，指定的连通性的类型不同，许多数学形态学操作的结果也会有所不同。

例如，如果指定一个 4 连通的邻域，下面二值图像包含两个对象；如果指定一个 8 连通的邻域，图像中没有对象。

```
0    0    0    0    0    0
0    1    1    0    0    0
0    1    1    0    0    0
0    0    0    1    1    0
0    0    0    1    1    0
```

通过指定一个元素为 0 和 1 的 3×3×…×3 的数组，还可以定义定制的邻域。值 1 相对于中心元素定义邻域的连通性。例如，下面的数组定义一个“North/South”连通性，它具有把图像分成独立的列的作用。

```
CONN = [0 1 0; 0 1 0; 0 1 0]
CONN =
     0    1    0
     0    1    0
     0    1    0
```

26.2.3 填充操作

imfill 函数对二值图像和灰度图像进行填充操作。对于二值图像，imfill 函数改变连接的背景像素和前景像素，到达对象边界时停止。对于灰度图像，imfill 函数将被亮区环绕的暗区的亮度值设置为与周围的相同。本操作对于从图像上删除不相关的人为缺陷很有帮助。

对于二值图像和灰度图像，填充操作的边界由指定的连通性确定。注意，imfill 函数与其他基于对象的操作不同的是，它对背景像素进行操作。用 imfill 函数指定连通性时，是指定背景的连通性，而不是前景的连通性。

连通性的含义可以用下面的矩阵表示。

```
BW = [0   0   0   0   0   0   0   0;
      0   1   1   1   1   1   0   0;
      0   1   0   0   0   1   0   0;
      0   1   0   0   0   1   0   0;
      0   1   0   0   0   1   0   0;
      0   1   1   1   1   0   0   0;
      0   0   0   0   0   0   0   0;
      0   0   0   0   0   0   0   0];
```

如果背景是 4 连通的，这幅二值图像包含两个单独的背景元素（回路以内的部分和回路以外的部分）。如果背景是 8 连通的，像素沿对角进行连接，并且只有一个背景元素。

对于二值图像，通过传入位置脚标或用交互模式使用 imfill 函数，可以指定填充操作的起点。例如，如果调用 imfill 函数，指定像素 BW(4,3)作为起点，imfill 函数只填充回路内部，因为默认时，背景是 4 连通的。

```
imfill(BW,[4 3])
ans =
   0   0   0   0   0   0   0   0
   0   1   1   1   1   1   0   0
   0   1   1   1   1   1   0   0
   0   1   1   1   1   1   0   0
   0   1   1   1   1   1   0   0
   0   1   1   1   1   0   0   0
   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0
```

如果指定相同的起点，但是使用 8 连通的背景连通性，imfill 函数将填充整幅图像。

```
imfill(BW,[4 3],8)
ans =
   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1
```

```
    1    1    1    1    1    1    1    1
    1    1    1    1    1    1    1    1
```

注式填充的一个常见用途是填充图像中的洞。例如，假设有一幅二值图像或灰度图像，其中前景对象表示球体，在图像中它们看起来应该像圆盘，但是因为原始照片中存在反射现象，现在看起来像圆环了。对该图像作进一步的处理以前，可以首先用 imfill 函数填充圆环中间的洞。

由于用注式填充填充洞非常常见，以至于 imfill 函数提供了特殊的语法来支持它填充二值图像和灰度图像。该语法中，只需要指定变量 "holes"，不必指定每个洞的起点。下面举例填充脊柱灰度图像中存在的洞。

```
[X,map] = imread('spine.tif');
I = ind2gray(X,map);
Ifill = imfill(I,'holes');
imshow(I);figure, imshow(Ifill)
```

生成图 26-10。



(a)　(b)

图 26-10　填充前后的图像

26.2.4　寻找峰和谷

灰度图像可以认为是三维的：x 和 y 轴表示像素位置，z 轴表示每个像素的亮度。按照这种解释，亮度值表示高度，就像在地形图中的那样。图像中的高亮度区域和低亮度区域，对应于地形学上的术语峰和谷，是重要的数学形态学特征，因为它们经常表示相关的图像对象。

例如，在一幅有几个球形对象的图像中，高亮度点可能表示对象的顶。使用数学形态学处理方法，这些最大值点可以用于识别图像中的对象。

本节包括以下内容：

- 理解 Maxima 和 Minima 函数；
- 寻找高亮度区或低亮度区；
- 抑制最小值和最大值；
- 强调显示最小值点。

1. 理解 Maxima 和 Minima 函数

一幅图像可以有多个局部最大值或最小值，但是只有一个全局最大值或最小值。确定图

像的峰或谷可以用于创建 Marker 图像，该图像用于数学形态学重建。

图 26-11 演示了一维情况下的概念。

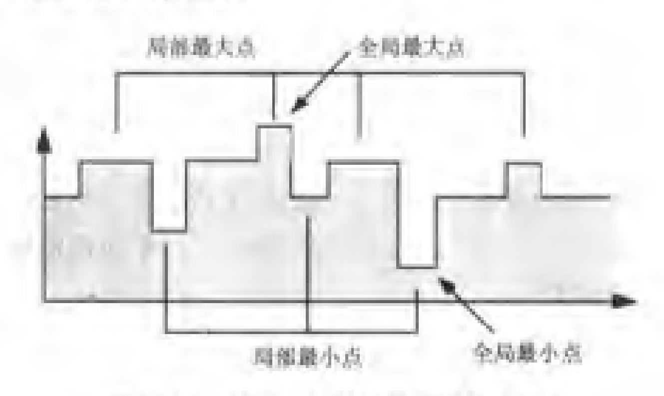


图 26-11　局部、全局的最大和最小值点

2．寻找高亮度区或低亮度区

工具箱中包含了可以用于查找图像中高亮度区或低亮度区的函数：

- imregionalmax 和 imregionalmin 函数确定所有局部最大值或最小值；
- imextendedmax 和 imextendedmin 函数确定所有大于或小于一个指定阈值的局部最大值或最小值。

这些函数把一幅灰度图像作为输入参数，返回一幅二值图像。在输出的二值图像中，局部最小值或最大值设置为 1，其他所有像素的值设置为 0。

例如，下面的简单图像包含两个主要的局部最大值，这些像素区块包含的值为 13 和 18，有几个更小的最大值，设置为 11。

```
A=[10 10 10 10 10 10 10 10 10 10;
   10 13 13 13 10 10 11 10 11 10;
   10 13 13 13 10 10 10 11 10 10;
   10 13 13 13 10 10 11 10 11 10;
   10 10 10 10 10 10 10 10 10 10;
   10 11 10 10 10 18 18 18 10 10;
   10 10 10 11 10 18 18 18 10 10;
   10 10 11 10 10 18 18 18 10 10;
   10 11 10 11 10 10 10 10 10 10;
   10 10 10 10 10 10 11 10 10 10];
```

imregionalmax 函数返回的二值图像可以找出所有这些局部最大值。

```
B = imregionalmax(A)
B =
     0     0     0     0     0     0     0     0     0     0
     0     1     1     1     0     0     1     0     1     0
     0     1     1     1     0     0     0     1     0     0
     0     1     1     1     0     0     1     0     1     0
     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     1     1     1     0     0
```

```
0    0    0    1    0    1    1    1    0    0
0    0    1    0    0    1    1    1    0    0
0    1    0    1    0    0    0    0    0    0
0    0    0    0    0    0    1    0    0    0
```

有时候可能只希望找出图像中那些亮度改变比较大的区域：即目标像素与相邻像素之间值的差异大于或小于一个给定阈值的区域。例如，如果只查找示例图像 A 中的那些局部最大值，并且要求这些局部最大值至少比它们的相邻像素值高两个单位，使用 imextendedmax 函数。

```
B = imextendedmax(A,2)
B =
    0    0    0    0    0    0    0    0    0    0
    0    1    1    1    0    0    0    0    0    0
    0    1    1    1    0    0    0    0    0    0
    0    1    1    1    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    1    1    1    0    0
    0    0    0    0    0    1    1    1    0    0
    0    0    0    0    0    1    1    1    0    0
    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0
```

3. 抑制最小值和最大值

图像中，亮度的每一个小的波动都表示一个局部的最小值或最大值。有时候可能只对比较明显的最小值或最大值感兴趣，对背景纹理引起的更小的最小值和最大值不感兴趣。使用 imhmax 或 imhmin 函数，在删除不明显的最小值和最大值的同时，保留明显的最小值和最大值。使用这些函数，可以指定一个对比准则或阈值水平 h，并利用它抑制所有高度小于 h 的最大值或高度大于 h 的最小值。

注意，imregionalmin,imregionalmax,imextendedmin 和 imextendedmax 函数返回一个二值图像，它标示出了图像中局部最小值和最大值的位置。imhmax 和 imhmin 函数生成一个改变后的图像。

例如，下面的简单图像包含两个主要的局部最大值——14 和 18，有几个小的最大值 11。

```
A=[10 10 10 10 10 10 10 10 10 10;
   10 14 14 14 10 10 11 10 11 10;
   10 14 14 14 10 10 10 11 10 10;
   10 14 14 14 10 10 11 10 11 10;
   10 10 10 10 10 10 10 10 10 10;
   10 11 10 10 10 18 18 18 10 10;
   10 10 10 11 10 18 18 18 10 10;
   10 10 11 10 10 18 18 18 10 10;
   10 11 10 11 10 10 10 10 10 10;
   10 10 10 10 10 10 11 10 10 10];
```

指定阈值 2，用 imhmax 函数剔除除了两个明显的最大值以外的所有局部最大值。注意，imhmax 函数只影响最大值，其他像素值不会发生改变。处理结果如下所示。可见，虽然两个明显最大值的高度减小了，但它们仍然保留下来。

```
B = imhmax(A,2)
B =
    10    10    10    10    10    10    10    10    10    10
    10    12    12    12    10    10    10    10    10    10
    10    12    12    12    10    10    10    10    10    10
    10    12    12    12    10    10    10    10    10    10
    10    10    10    10    10    10    10    10    10    10
    10    10    10    10    10    16    16    16    10    10
    10    10    10    10    10    16    16    16    10    10
    10    10    10    10    10    16    16    16    10    10
    10    10    10    10    10    10    10    10    10    10
    10    10    10    10    10    10    10    10    10    10
```

4. 强调显示最小值点

可以用 imimposemin 函数强调显示图像中指定的最小值点。该函数用数学形态学重建来剔除图像中除指定最小值以外的所有最小值。

例如，下面的代码创建一个包含两个主要的局部最小值和少数几个其他局部最小值的简单图像。

```
Mask = uint8(10*ones(10,10));
Mask(6:8,6:8) = 2;
Mask(2:4,2:4) = 7;
Mask(3,3) = 5;
Mask(2,9) = 9
Mask(3,8) = 9
Mask(9,2) = 9
Mask(8,3) = 9
Mask =
    10    10    10    10    10    10    10    10    10    10
    10     7     7     7    10    10    10    10     9    10
    10     7     5     7    10    10    10    10    10    10
    10     7     7     7    10    10    10    10    10    10
    10    10    10    10    10    10    10    10    10    10
    10    10    10    10    10     2     2     2    10    10
    10    10    10    10    10     2     2     2    10    10
    10    10    10    10    10     2     2     2    10    10
    10    10    10    10    10    10    10    10    10    10
    10    10    10    10    10    10    10    10    10    10
```

要想获得一幅只强调显示两个最小的局部最小值并且删除所有其他局部最小值的图

像，需要创建一幅标明了感兴趣的两个局部最小值的 Marker 图像。可以通过将特定像素明确设置为指定值来创建 Marker 图像，或者使用其他数学形态学函数来提取希望在 Mask 图像中强调的特征。

下面的例子用 imextendedmin 函数获取一幅二值图像，它显示了两个最小最小值（指局部最小值，后同）的位置。

```
Marker = imextendedmin(Mask,1)
Marker =
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     1     1     0     0
     0     0     0     0     0     1     1     1     0     0
     0     0     0     0     0     1     1     1     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```

现在用 imimposemin 函数在 Mask 图像中，在 Marker 图像指定的点上创建新的最小值点。注意 imimposemin 函数是如何将 Marker 图像指定的像素的值设置为数据类型支持的最小值的。imimposemin 函数还改变图像中所有其他像素的值，以便剔除其他最小值。

```
I = imimposemin(Mask,Marker)
I =
    11    11    11    11    11    11    11    11    11    11
    11     8     8     8    11    11    11    11    11    11
    11     8     0     8    11    11    11    11    11    11
    11     8     8     8    11    11    11    11    11    11
    11    11    11    11    11    11    11    11    11    11
    11    11    11    11    11     0     0     0    11    11
    11    11    11    11    11     0     0     0    11    11
    11    11    11    11    11     0     0     0    11    11
    11    11    11    11    11    11    11    11    11    11
    11    11    11    11    11    11    11    11    11    11
```

26.3 距离变换

距离变换提供了图像中离散点之间的一个度量标准。图像处理工具箱中的 bwdist 函数计算二值图像中设置为 off(0)的像素与其最近非 0 像素之间的距离。

bwdist 函数支持几种距离度量标准，如表 26-5 中所示。

表 26-5　距离矩阵

距离度量	描　述	演　示
Euclidean	两个像素之间的直线距离	图像　距离变换
City Block	基于 4 连通邻域度量像素之间的路径，共边像素的距离为 1 个单位；对角接触的像素距离为 2 个单位	图像　距离变换
Chessboard	基于 8 连通邻域进行度量，共边或共顶点的像素之间的距离为 1 个单位	图像　距离变换
Quasi-Euclidean	沿一系列水平、垂直和对角线段度量总的欧拉距离	图像　距离变换

下面的例子创建一个包含两个相交圆形对象的二值图像。

```
center1 = -10;
center2 = -center1;
dist = sqrt(2*(2*center1)^2);
radius = dist/2 * 1.4;
lims = [floor(center1-1.2*radius) ceil(center2+1.2*radius)];
[x,y] = meshgrid(lims(1):lims(2));
bw1 = sqrt((x-center1).^2 + (y-center1).^2) <= radius;
bw2 = sqrt((x-center2).^2 + (y-center2).^2) <= radius;
bw = bw1 | bw2;
figure, imshow(bw)
```

结果如图 26-12 所示。

用 bwdist 函数进行二值图像的距离变换计算。在距离变换图像中，注意两个圆形区域的中心是如何变成白色的。

```
D = bwdist(~bw);
figure, imshow(D,[])
```

生成图 26-13。



图 26-12　两个相交圆图像的二值图像

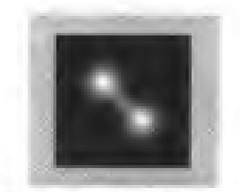


图 26-13　距离变换结果

26.4　对象、区域和特征度量

工具箱包括几个返回与二值图像特征有关信息的函数，包括：

- 连接组分的标注，并且利用标注矩阵获取与图像有关的统计量；
- 在二值图像中选择对象；
- 找到二值图像的前景区域；
- 找到二值图像的欧拉数。

26.4.1　连接组分的标注

bwlabel 函数和 **bwlabeln** 函数进行连接组分的标注，标注以后，就可以识别二值图像中的每一个对象。**bwlabel** 函数只支持二维的输入，**belabeln** 函数支持任何维数的输入。

这些函数返回一个称为标注矩阵的矩阵。一个标注矩阵就是一幅图像，它的大小与输入图像相同，输入图像中的对象通过输出矩阵中的不同整型值来进行区分。例如，**bwlabel** 函数可以识别这幅二值图像中的对象。

```
BW = [0    0    0    0    0    0    0    0;
      0    1    1    0    0    1    1    1;
      0    1    1    0    0    0    1    1;
      0    1    1    0    0    0    0    0;
      0    0    0    1    1    0    0    0;
      0    0    0    1    1    0    0    0;
      0    0    0    1    1    0    0    0;
      0    0    0    0    0    0    0    0];
X = bwlabel(BW,4)
X =
      0    0    0    0    0    0    0    0
      0    1    1    0    0    3    3    3
      0    1    1    0    0    0    3    3
      0    1    1    0    0    0    0    0
      0    0    0    2    2    0    0    0
      0    0    0    2    2    0    0    0
      0    0    0    2    2    0    0    0
      0    0    0    0    0    0    0    0
```

在输出矩阵中，1 组成的区域表示第 1 个对象，2 组成的区域表示第 2 个对象，3 组成的区域表示第 3 个对象。

26.4.2　查看标注矩阵

bwlabel 或 bwlabeln 函数返回的标注矩阵是 double 型的，它不是二值图像。进行查看的方法之一是用 label2rgb 函数把它显示成假彩色索引图像。在假彩色图像中，标注矩阵中识别每个对象的数字作为索引值与颜色映射矩阵相连。把标注矩阵看作 RGB 图像时，图像中

的对象更容易分辨。

为了演示这个技巧，下面的例子用 label2rgb 函数查看标注矩阵 ***X***。调用 label2rgb 函数时指定了标准 MATLAB 映射矩阵之一 jet。第 3 个变量 *k* 指定背景色。

```
BW1 = [0    0    0    0    0    0    0    0;
       0    1    1    0    0    1    1    1;
       0    1    1    0    0    0    1    1;
       0    1    1    0    0    0    0    0;
       0    0    0    1    1    0    0    0;
       0    0    0    1    1    0    0    0;
       0    0    0    1    1    0    0    0;
       0    0    0    0    0    0    0    0];
X = bwlabel(BW1,4);
RGB = label2rgb(X, @jet, 'k');
imshow(RGB,'notruesize')
```

生成图 26-14。



图 26-14　标注矩阵的图像

26.4.3 计算二值图像中前景的面积

bwarea 函数返回二值图像的面积。该面积是对图像中前景大小的一个度量。粗略地讲，该面积是图像中设置为 on 的像素个数。但是 bwarea 函数并不简单地计算设置为 on 的像素的个数。更准确地说，bwarea 函数计算面积时对不同像素的加权处理是不同的。这种加权处理可以补偿图像失真带来的遗憾。而图像失真是固有的，用离散像素表示连续图像就会导致失真。例如，一条 50 像素的对角线比一条 50 像素的水平线长。bwarea 函数加权处理以后的结果是，水平线的面积是 50，而对角线的面积是 62.5。

下面的例子用 bwarea 函数计算图像 circbw.tif 经过膨胀处理以后面积增长的百分比。

```
BW = imread('circbw.tif');
SE = ones(5);
BW2 = imdilate(BW,SE);
increase = (bwarea(BW2) - bwarea(BW))/bwarea(BW);
increase =
    0.3456
```

26.4.4 计算二值图像中的欧拉数

bweuler 函数返回二值图像的欧拉数。欧拉数是对图像的拓扑度量。它指的是图像中对象的总个数与这些对象中洞的个数的差。可以使用 4 连通或 8 连通邻域。

下面的例子计算电路图的欧拉数，使用了 8 连通邻域。

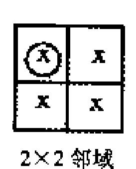
```
BW1 = imread('circbw.tif');
eul = bweuler(BW1,8)
eul =
    -85
```

该例中，欧拉数是负的，表示洞的个数比对象的个数还要多。

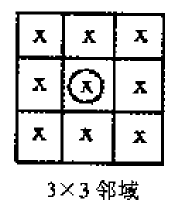
26.5 调查表

有些二值图像处理可以很简单地通过调查表实现。调查表是一个列向量，其中每一个元素表示相邻像素的一种可能组合。

可以用 makelut 函数为不同操作创建调查表。makelut 函数为 2×2 和 3×3 的邻域创建调查表。图 26-15 演示了这些邻域类型。*x* 表示相邻像素，打圆圈了的像素为中心像素。



2×2 邻域



3×3 邻域

图 26-15　两种邻域类型

对于 2×2 的邻域，相邻像素间有 16 种可能的排列关系。所以，本操作的调查表是一个 16 元素的向量。对于 3×3 的邻域，有 512 种排列关系。所以调查表是一个有 512 个元素的向量。

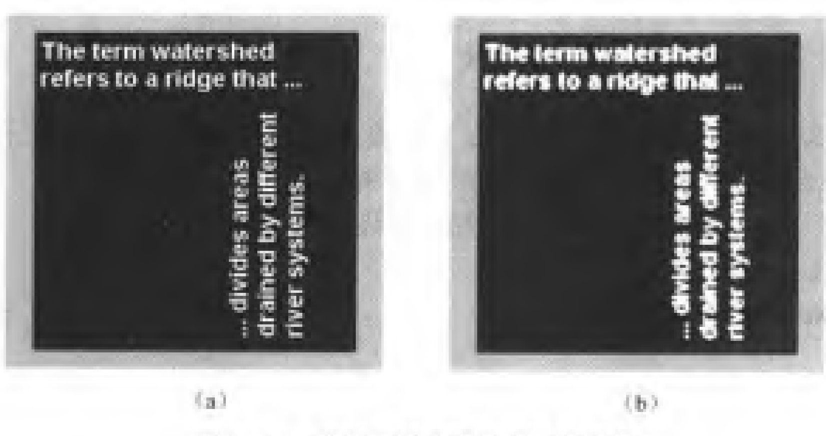
一旦创建了调查表，就可以通过调用 applylut 函数，用它完成必要的操作。下面的例子演示如何用调查表操作修改一个包含文本的图像。首先，编写一个函数，如果 3×3 的相邻像素中有 3 个以上的像素值为 1，则返回 1，否则返回 0。然后调用 makelut 函数，把该返回值作为第 1 个变量传递给它，并用第 2 个变量指定一个 3×3 的调查表。

```
f = inline('sum(x(:)) >= 3');
lut = makelut(f,3);
```

lut 是一个 512 元素的向量，元素值为 1 或 0。然后，用 applylut 函数完成操作。

```
BW1 = imread('text.png');
BW2 = applylut(BW1,lut);
imshow(BW1)
figure, imshow(BW2)
```

生成图 26-16，其中图（a）和图（b）分别为进行查找表操作前后的图像。



（a）　　（b）

图 26-16　进行查找表操作前后的图像

注意，不能将 makelut 函数和 applylut 函数用于大于 2×2 或 3×3 的邻域。这些函数只支持 2×2 或 3×3 的邻域，因为调查表对于大于 3×3 的邻域是不实用的。

第 27 章　图 像 分 析

MATLAB 图像处理工具箱提供了灰度统计、直方图、等值线图和边缘检测、边界跟踪四叉树分解等图像分析功能。

27.1　像素值和统计量

图像处理工具箱提供了几个函数来返回与图像数据有关的信息。这些函数用不同形式返回这些信息，包括：

- 选定点的数据值（pixval,impixel 函数）；
- 图像中沿一定路径分布的数据值（improfile 函数）；
- 图像数据的等值线图（imcontour 函数）；
- 图像数据的直方图（imhist 函数）；
- 图像数据的综述统计量（mean2,std2 和 corr2 函数）；
- 图像区域的特征度量（regionprops 函数）。

27.1.1　像素选择

工具箱中有两个函数提供与图像中指定像素的颜色数据值有关的信息。

● pixval 函数可以交互显示图像上像素的数据值。它还可以显示两个像素之间的欧拉距离。

● impixel 函数返回选定的一个或一系列像素的数据值。可以将选定像素的坐标作为输入变量，或者可以用鼠标选择像素。

注意，对于索引图像，pixval 函数和 impixel 函数都显示保存在颜色映射矩阵中的 RGB 值，而不是索引值。

使用 pixval 函数时，首先显示一幅图像，然后输入 pixval 命令。pixval 函数在图像下面放一个黑条，这个黑条可以显示鼠标下方像素的坐标值(x, y)和该像素的颜色数据。

如果用鼠标在图像上单击以后进行拖拉，则 pixval 函数还会显示单击点与鼠标当前点之间的欧拉距离。pixval 函数通过在这些点之间画线来表示它们之间的距离已经测量了。释放鼠标时，直线和距离显示消失。

pixval 函数即时给出的结果比 impixel 函数给出的多，但是使用 impixel 函数有用变量返回结果的好处，这个变量可以以交互或非交互的方式被调用。如果以无参方式调用 impixel 函数，则鼠标位于图像上方时光标变成十字形。然后可以单击感兴趣的像素，impixel 函数在选择的每个像素上显示一个小的星形。选定以后，单击回车键，impixel 函数返回选定像素的颜色值，然后星形消失。

下面的例子演示了 impixel 函数的使用。

（1）显示图像。

```
imshow canoe.tif
```

（2）调用 impixel 函数选择像素点。

```
vals = impixel
```

（3）在图像上单击一些点来选择像素，完成以后，单击回车键，如图 27-1 所示。



图 27-1　在图像上选择像素

impixel 函数将像素值返回到 vals 中。

```
vals =
    0.5490    0.4510    0.2588
    0.5490    0.5804    0.6118
    0.2588    0.2235    0.1922
```

27.1.2　灰度轮廓

improfile 函数沿图像中的线段或多义线计算和显示灰度。可以将直线段的端点坐标作为输入变量，也可以用鼠标定义路径。使用第 2 种方法时，improfile 函数使用插值法来确定路径上等间隔点的值。（默认时，improfile 函数使用最近邻插值，但是可以指定其他不同的方法。）improfile 函数用于灰度图像和 RGB 图像时效果最佳。

如果只有一条直线段，则 improfile 函数用二维图显示灰度值。如果是多义线，则 improfile 函数用三维图显示灰度值。

如果调用无参的 improfile 函数，则鼠标光标位于图像上方时变为十字形。然后可以通过单击线段的终点来定义线段；improfile 函数会在两个连续选定的点之间进行连线。指定完路径以后，单击回车键，improfile 函数将图形显示在一个新的图形窗口中。

下面的例子调用 improfile 函数并用鼠标指定一条直线段。

```
I = fitsread('solarspectra.fts');
imshow(I,[]);
improfile
```

效果如图 27-2 所示。图中，直线段的位置用红色表示，从上至下绘制。

improfile 函数显示沿直线分布的数据值，如图27-3 所示。注意线形图中的峰和谷是如何与图像上的亮带和黑带相对应的。

下面的例子显示了 improfile 函数是如何用于RGB 图像的。使用 imshow 函数可以在图形窗口中显示图像。调用不带参数的 improfile 函数，在图像上交互式地跟踪直线段。

```
imshow peppers.png
improfile
```

在生成的图中，用鼠标画直线段，双击鼠标，终止绘制，见图 27-4。图中黑线表示用鼠标画的线。

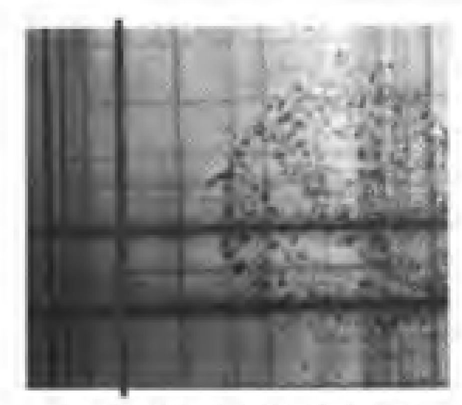


图 27-2 在图像上绘一条直线段

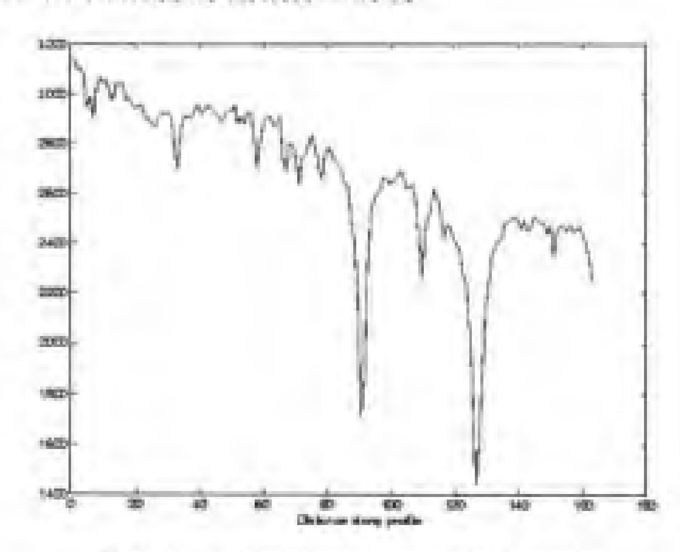


图 27-3 用线形图表示沿直线分布的数据值



图 27-4 在图像上绘一条直线段

improfile 函数显示沿直线段分布的 RGB 值的线形图，如图 27-5 所示。图像中分别包括了红色、绿色和蓝色亮度值的图形。

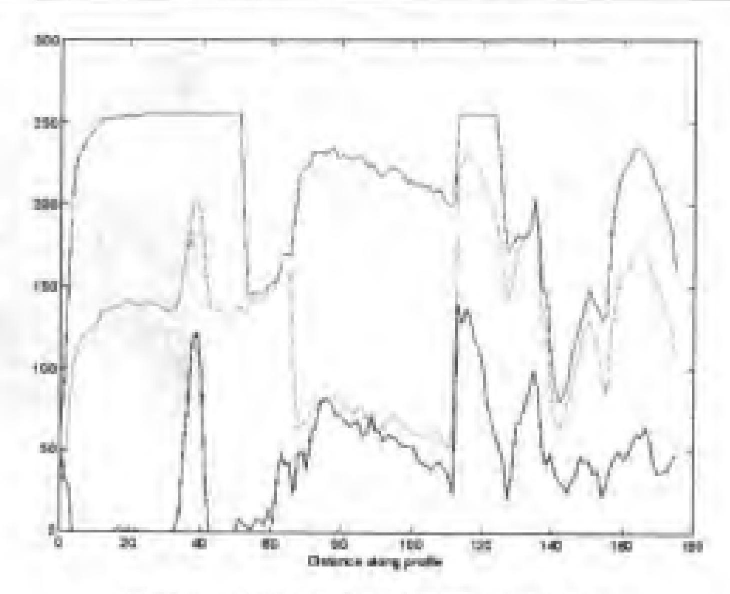


图 27-5　沿直线段分布的 RGB 值的线形图

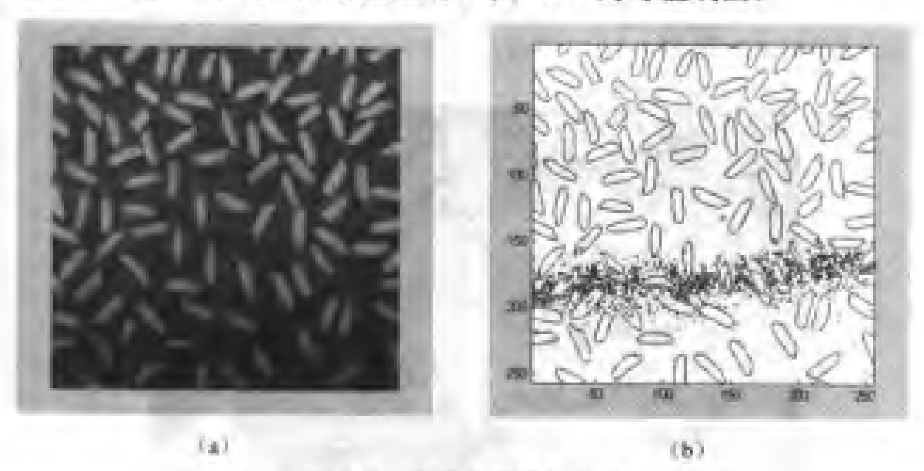
27.1.3　图形等值线

可以用 imcontour 函数显示灰度图中数据的等值线图。该函数与 MATLAB 中的 contour 函数相似，但它会自动设置坐标轴，使得它们的方向和大小比例与图像匹配。

下面的例子显示米粒灰度图像和它的等值线图。

```
I = imread('rice.png');
imshow(I)
figure, imcontour(I,3)
```

如图 27-6 所示，图（a）为米粒灰度图像，图（b）为等值线图。



（a）　　（b）

图 27-6　图像及其等值线图

可以用 clabel 函数标注等值线的水平。

27.1.4　图像直方图

图像等值线是一种显示索引图像或灰度图像亮度分布的图形。用 imhist 函数创建图像

直方图。该图首先将数据分成 n 个等间距的条形，每个条形表示一个数据范围，然后计算落在这个范围内像素的个数。

例如，下面的命令显示一个米粒图像和一个有 64 个条形的直方图。该直方图显示了一个值为 100 左右的峰，对应于图像背景中的深灰色。

```
I = imread('rice.png');
imshow(I)
figure, imhist(I)
```

如图 27-7 所示。


(a)

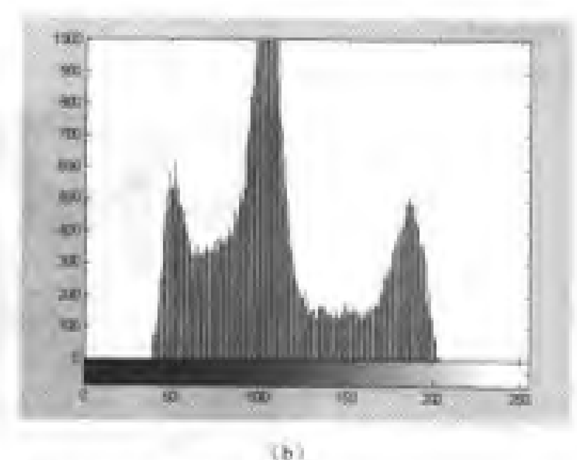

(b)

图 27-7 图像及其直方图

27.1.5 综述统计量

可以用 mean2, std2 和 corr2 函数计算图像的标准统计量。mean2 和 std2 函数计算矩阵中元素的均值和标准差。corr2 函数计算两个大小相同的矩阵的相关系数。

27.1.6 区域属性度量

可以用 regionprops 函数计算图像区域的属性。例如，regionprops 函数可以度量诸如面积、质心和包围盒等属性。

27.2 边缘检测

可以用 edge 函数进行边缘检测。这里的边缘指的是图像中对象的边界。为了进行边缘检测，edge 函数使用下面两个准则中的一种，在图像上查找亮度剧烈改变的地方。

- 亮度的一阶导数在量级上比某些阈值更大的地方；
- 亮度的二阶导数为 0 的地方。

edge 函数提供了多个求导器，每个求导器可以解决上面问题中的一种。对于有些求导器，可以指定操作是否对水平边界、垂直边界或两者都敏感。edge 函数返回一个二值图像，图像中找到了边界的像素用 1 表示，否则用 0 表示。

edge 函数提供的边缘检测方法中，最有力的是 Canny 法。该方法与其他边缘检测方法的主要区别在于，它使用两个阈值来检测强边界和弱边界，并且只在弱边界与强边界相连时才进行显示。所以，该方法与其他方法相比，受噪声影响的机会更小，并且更有可能找到真实的弱边界。

下面的例子通过比较 Sobel 边缘检测器和 Canny 边缘检测器在同一图像上的应用效果来演示后者的能力。

（1）读入图像并显示它。

```
I = imread('coins.png');
imshow(I)
```

显示图 27-8。



图 27-8　原图像

（2）对图像应用 Sobel 和 Canny 边缘检测器并显示结果。

```
BW1 = edge(I,'sobel');
BW2 = edge(I,'canny');
imshow(BW1)
figure, imshow(BW2)
```

生成图 27-9 和图 27-10。

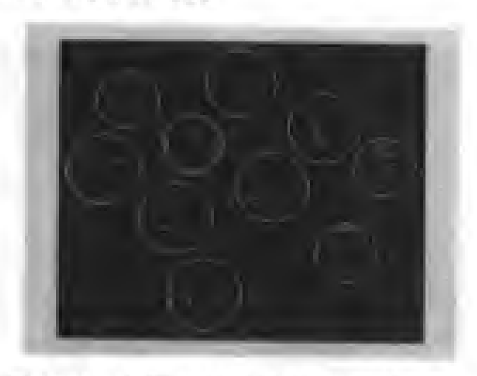


图 27-9　应用 Sobel 边缘检测器的效果

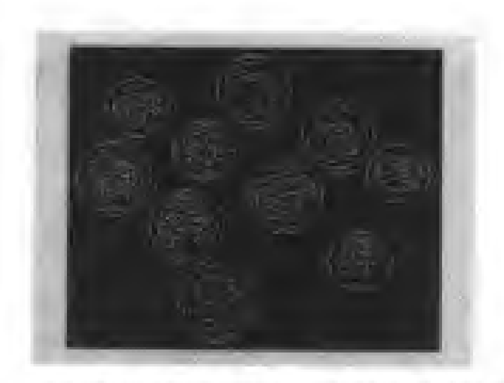


图 27-10　应用 Canny 边缘检测器的效果

27.3　边界跟踪

工具箱提供了两个函数，可以利用它们查找二值图像中的对象边界。这两个函数是

bwtraceboundary 函数和 bwboundaries 函数。

bwtraceboundary 函数图像中对象边界上所有像素的行坐标和列坐标，必须指定对象上一个边界像素的位置作为跟踪的起点。bwboundaries 函数返回图像中所有对象边界像素的行坐标和列坐标。

对于这两个函数，非 0 像素属于对象，值为 0 的像素组成背景。

下面的例子使用 bwtraceboundary 函数跟踪二值图像中的对象边界，然后用 bwboundaries 函数获取图像中所有对象的边界像素坐标。

（1）读入图像并显示它。

```
I = imread('coins.png');
imshow(I)
```

生成图 27-11。

（2）将图像转换为二值图像。bwtraceboundary 函数和 bwboundaries 函数都只用于二值图像。

```
BW = im2bw(I);
imview(BW)
```

生成图 27-12。



图 27-11　原图像



图 27-12　转换为二值图像

（3）确定要跟踪的对象边界上一个像素的行坐标和列坐标。bwboundary 函数把该点作为边界跟踪的起点。

```
dim = size(BW)
col = round(dim(2)/2)-90;
row = min(find(BW(:,col)))
```

（4）调用 bwtraceboundary 函数，从指定点开始跟踪边界。作为必需的变量，必须指定一个二值图像，起点的行坐标和列坐标，以及第 1 步的方向。本例中将北作为第 1 步的方向。

```
boundary = bwtraceboundary(BW,[row, col],'N');
```

（5）显示原始灰度图像，然后利用 bwtraceboundary 函数返回的坐标显示边界。

```
imshow(I)
hold on;
plot(boundary(:,2),boundary(:,1),'g','LineWidth',2);
```

生成图 27-13。



图 27-13　带有跟踪边的图像

（6）调用 bwboundaries 函数跟踪图像中所有硬币的边界。默认时，bwboundaries 函数查找图像中所有对象的边界，包括位于其他对象内部的对象。在本例使用的二值图像中，有些硬币包含黑色的区域，bwboundaries 函数将它们当作单独的对象。为了确保 bwboundaries 函数只返回这 10 个硬币的边界，本例指定“noholes”变量。

```
boundaries = bwboundaries(filled,'noholes');
```

（7）正如下面输出中所显示的，bwboundaries 函数返回图像中所有硬币边缘像素的坐标值。

```
whos boundaries
  Name        Size                Bytes  Class
  boundaries  10x1                24 936  cell array
```

对于某些对象，选择起点和方向时必须小心。例如，如果对象包含有洞并且起点位于对象的狭长部分，根据第 1 步选择的方向的不同，可能会跟踪到对象边界以外或者洞边界的内部。对于填充了的对象，第 1 步选择的方向没有这么重要。

27.4　四叉树分解

四叉树分解是一种图像分析技术，它将图像二次分解为比图像本身更均一的块。该技术揭示了图像的结构信息。它还可以用于适应性压缩算法的第 1 步。

可以用 qtdecomp 函数进行四叉树分解。该函数将一个方形图像分成 4 个大小相同的块，然后测试每一个块，看它是否符合某些均一性（即块内像素是否处于指定的动态范围内）准则。如果块符合准则，则不作进一步的分解。如果不符合准则，将该块再分解为 4 个小块，然后测试其他块。重复这个过程，直到每个块都符合准则。结果可能会有几种不同大小的块。

例如，假设想对一个 128×128 的灰度图像进行四叉树分解。第 1 步是将图像分成 4 个 64×64 的块，然后对每个块应用测试标准。例如，准则可以是下面这样：

```
max(block(:)) - min(block(:)) <= 0.2
```

如果一个块符合这个准则，则不作进一步的分解，64×64 是最终的分解结果。如果块不符合准则，则将该块进一步分解为 32×32 的块，然后对其中的每个块进行测试。测试失败的块又进一步分解为 4 个 16×16 的块，依此类推，直到所有块都通过测试。除非另外指

定，有些块可以小到1×1。

调用qtdecomp函数进行四叉树分解，把图像和阈值作为变量。

```
S = qtdecomp(I,0.27)
```

不管图像I是什么类型的，把阈值指定为0和1之间的数。如果I是uint8型的，则qtdecomp函数用255乘以阈值来确定实际使用的阈值。如果I是uint16型的，则qtdecomp函数用65 535乘以阈值。*S*作为稀疏矩阵返回，大小与I的相同。*S*中的非0元素表示块的左上角，每一个非0元素的值表示块的大小。

图27-14显示了一幅图像和它的四叉树分解表示。每一个黑色方形表示一个均一的块，白线表示块之间的边界。

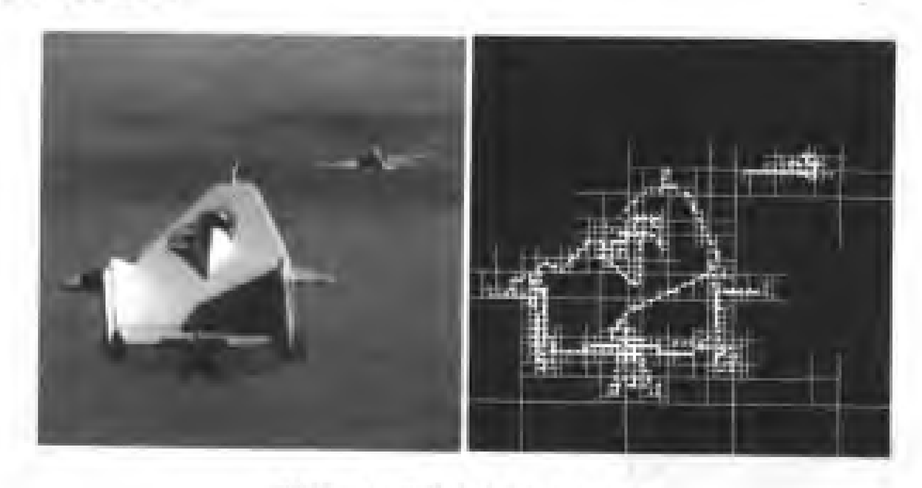


图27-14 图像和它的四叉树表示

第28章 图像增强

图像增强常用于改善图像质量。这里的“改善”，有客观方面的改善，如增加信号和噪声比率；也有主观方面的改善，如通过改变颜色或灰度，使图像的某些特征更容易识别。

28.1 灰度调整

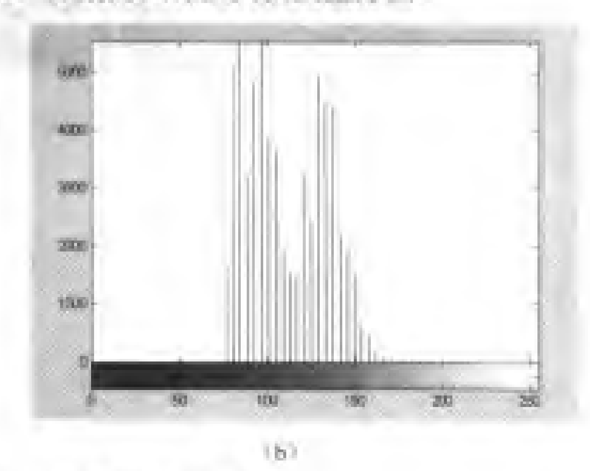
灰度调整是一种图像增强技术，它将图像的灰度值映射到一个新的范围。为了演示，图28-1用直方图显示了一幅灰度对比度比较低的图像。在命令窗口键入下面的代码。

```
I = imread('pout.tif');
imshow(I)
figure, imhist(I,64)
```

生成图28-1，其中图（a）和图（b）分别为图像及其灰度直方图。



（a）



（b）

图28-1 图像及其直方图

如果将数据值重新映射到整个灰度范围[0, 255]，则可以增加图像的对比度。下面介绍几种灰度调整技术，包括：

- 将灰度值调整到一个指定的范围；
- 直方均等化；
- 有限对比适应性直方均等化；
- 去相关拉伸。

这一节介绍的函数只适用于灰度图像。但是，有些函数也适用于彩色图像。

28.1.1 将灰度值调整到一个指定的范围

可以用 imadjust 函数将图像的灰度值调整到一个指定的范围。例如，下面的代码通过将

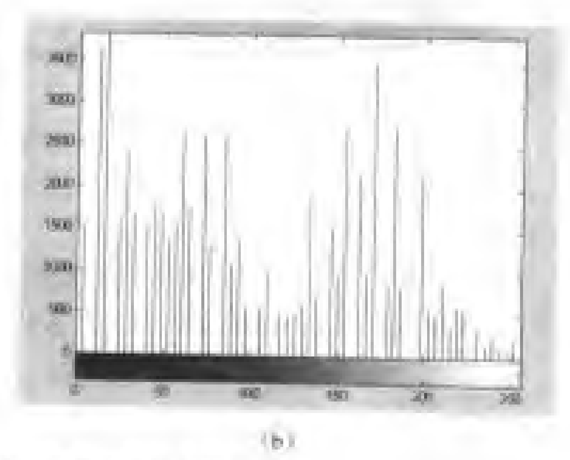
一幅对比度比较低的灰度图像的数据值映射到整个灰度范围[0, 255]来增加图像的对比度。

```
I = imread('pout.tif');
J = imadjust(I);
imshow(J)
figure, imhist(J,64)
```

图 28-2（a）和（b）显示了调整后的图像和它的直方图。注意，现在图像的对比度增加了，而且直方图填充到整个区域。



（a）



（b）

图 28-2　调整图像的灰度值

1. 指定调整限制

可以用 imadjust 函数有选择地指定输入值和输出值的范围。将这些范围放在两个向量中，然后把这两个向量作为变量传递给 imadjust 函数。第 1 个向量指定要映射的低灰度值和高灰度值，第 2 个向量指定进行映射的比例。

例如，通过收缩数据范围可以减小图像的对比度。图 28-3 中，摄影者的大衣太黑了，无法查看衣服上的细节。imadjust 函数将 uint8 型输入图像上的灰度数据范围[0, 51]映射到输出图像的[128, 255]。该操作在一定程度上加亮了图像，并且拓宽了原图像中深色区域的动态范围。

```
I = imread('cameraman.tif');
J = imadjust(I,[0 0.2],[0.5 1]);
imshow(I)
figure, imshow(J)
```

生成图 28-3，其中图（a）和图（b）分别为处理前后的图像。

2. 自动设置调整限制

使用 imadjust 函数，必须完成下面两个步骤：

（1）查看图像的直方图，确定图像的灰度限制；

（2）将这些限制指定为 0.0 和 1.0 之间的小数，这样，可以将它们放在[low_in high_in]向量中传递给 imadjust 函数。

用 stretchlim 函数，可以更方便地指定这些限制。该函数计算图像的直方图并自动确定调整限制。stretchlim 函数把这些值作为小数返回到一个向量中，可以把它作为[low_in high_in]变量传递给 imadjust 函数，例如，




（a）　　　　（b）

图 28-3　拓宽深色区域显示的动态范围

```
I = imread('rice.png');
J = imadjust(I,stretchlim(I),[0 1]);
```

默认时，stretchlim 函数将图像灰度范围内最低 1%和最高 1%的灰度值作为调整限制。通过清除灰度范围两端的极值，stretchlim 函数为剩余的灰度在动态调整范围内提供了更多余地。

3. Gamma 校正

imadjust 函数将 low 和 high 的值分别映射到输出图像的最低和最高灰度值。默认时，这种映射关系是线性的。例如，low 和 high 中间的值对应于输出图像最低值和最高值中间的那个值。

imadjust 函数还接受一个附加变量来指定 gamma 矫正因子。gamma 的值不同，输入图像和输出图像二者值之间的映射可能是非线性的。例如，low 和 high 中间的值映射到输出图像中的那个值可能会大于或小于输出图像最低值和最高值中间的那个值。

gamma 可以是 0 和无限值之间的任何值。如果 gamma 为 1（默认时），则映射是线性的。如果 gamma 小于 1，则映射会加权至更高的输出值。如果 gamma 大于 1，则映射会加权至更小的输出值。

下面的例子演示了 gamma 校正。注意，调用 imadjust 函数时，输入图像和输出图像的数据范围指定为空矩阵。指定一个空矩阵时，imadjust 函数使用默认的范围[0, 1]。本例中，两个范围都是空的，表示在没有其他数据调整的情况下使用 gamma 校正。

```
[X,map] = imread('forest.tif')
I = ind2gray(X,map);
J = imadjust(I,[],[],0.5);
imshow(I)
figure, imshow(J)
```

生成图 28-4，图（a）和图（b）分别为进行 gamma 校正前后的图像。

28.1.2　直方均等化

调整灰度值的处理可以用 histeq 函数自动完成，该函数进行直方均等化。所谓直方均等化，指的是转换图像的灰度值，使得输出图像的直方图与指定的直方图近似匹配。默认时，histeq 函数用一个有 64 个条块的直方图进行匹配。



（a）



（b）

图 28-4 进行 gamma 校正前后的图像

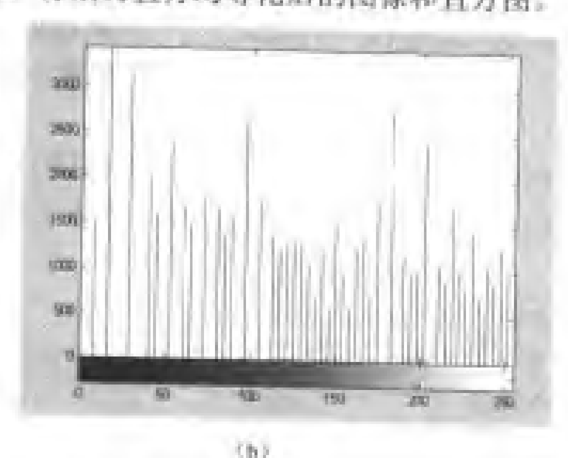
下面的例子演示如何用 histeq 函数调整灰度图。原图像的灰度对比度较低，大部分值位于灰度范围的中间。histeq 函数生成一幅灰度值在整个范围内均匀分布的输出图像。

```
I = imread('pout.tif');
J = histeq(I);
imshow(J)
figure, imhist(J,64)
```

生成图 28-5，其中图（a）和图（b）分别为直方均等化后的图像和直方图。



（a）



（b）

图 28-5 对图像进行直方均等化

histeq 函数返回一个 1×256 的向量，它为每个可能的输入值显示生成的输出值。不管输入图像是什么类型，向量中的值都落在[0, 1]范围内。可以利用这些数据画图，获得转换曲线。例如，

```
I = imread('pout.tif');
[J,T] = histeq(I);
figure,plot((0:255)/255,T);
```

生成图 28-6。

注意图 28-6 中的曲线与反映图 28-5 中灰度的直方图是对应的，输入值大部分落在 0.3 到 0.6 之间，输出值在 0 和 1 之间均匀分布。

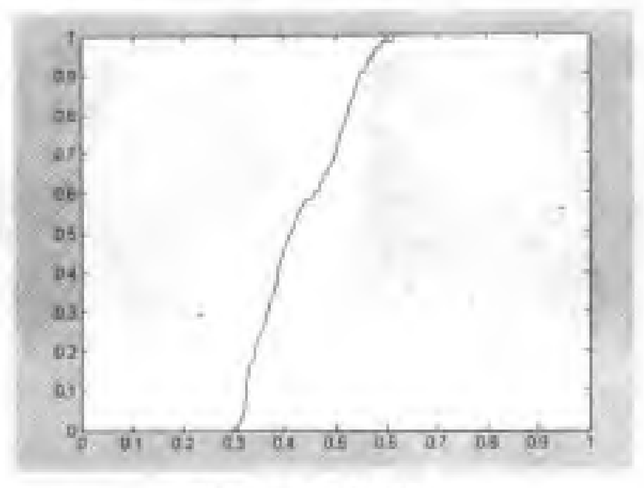


图 28-6　数据的线形图

28.1.3　有限对比适应性直方均等化

作为 histeq 函数的替换方法，可以用 adapthisteq 函数进行有限对比适应性直方均等化。二者的主要区别在于，histeq 函数处理整个图像，而 adapthisteq 函数处理图像中的小区域。每一个小区域的灰度对比增强了，所以输出区域的直方图与指定直方图近似匹配。进行均等化以后，adapthisteq 函数用双线性插值的方法来组合相邻的小区域，以剔除人为生成的边界。

为了避免放大可能存在于图像中的任何噪声，可以使用 adapthisteq 可选参数来限制对比度，对于均一区域尤其如此。

下面的例子用 adapthisteq 函数调整灰度图像中的对比度。原图像灰度对比度比较低，大部分值落在灰度范围的中间。adapthisteq 函数生成一幅灰度值在整个范围内均匀分布的输出图像。

```
I = imread('pout.tif');
J = adapthisteq(I);
imshow(I)
figure, imshow(J)
```

生成图 28-7，其中图（a）和图（b）分别为调整灰度对比度前后的图像。



（a）



（b）

图 28-7　调整图像灰度的对比度

28.1.4 去相关拉伸

去相关拉伸用紧密带-带相关来增强图像的颜色分离。这些超常的颜色改善了图像的可视化解释，并使图像的特征分离更容易。用 decorrestretch 函数进行去相关分离。

图像中色带的数目 NBANDS 通常为 3，但是应用去相关拉伸方法时可以不管色带的数目。图像原来的颜色值已经映射到一个新的颜色值系列，并且范围更宽。每个像素的颜色亮度被转换到 NBANDS 方差或相关矩阵的颜色特征空间，拉伸为等带方差，然后转换回原来的色带。

1. 简单的去相关拉伸

可以对 imdemos 目录下图像库中的图像进行去相关和拉伸操作。该图像库中包含有美国小科罗拉多河的地球资源卫星图像。下面对该图像进行简单的去相关拉伸操作。

（1）图像有 7 个带，但是这里只读入 3 种可见的颜色。

```
A = multibandread('littlecoriver.lan', [512, 512, 7], ...
'uint8=>uint8', 128, 'bil', 'ieee-le', ...
{'Band','Direct',[3 2 1]});
```

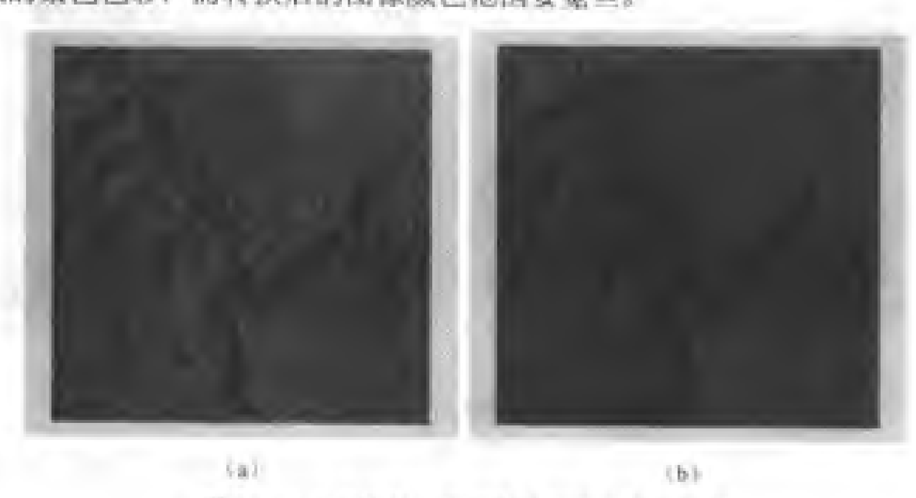
（2）然后进行去相关拉伸。

```
B = decorrstretch(A);
```

（3）查看结果。

```
imshow(A); figure; imshow(B)
```

如图 28-8 所示，其中图（a）和（b）为去相关拉伸前、后的图像。比较两幅图像，原图带有更强的紫色色彩，而转换后的图像颜色范围要宽些。



（a） （b）

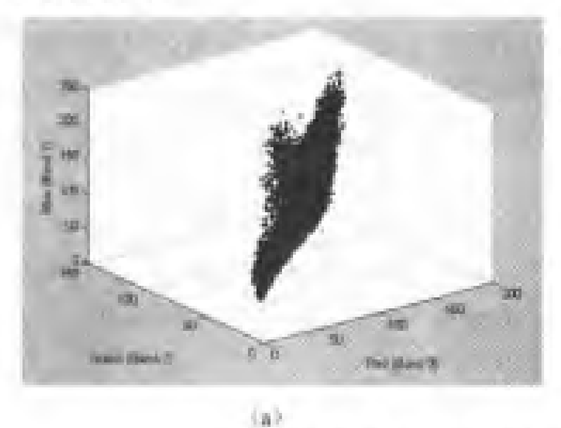
图 28-8 去相关拉伸前后的小科罗拉多河图像

下面生成的图像色带散点图显示了色带是如何被去相关和均一化的。

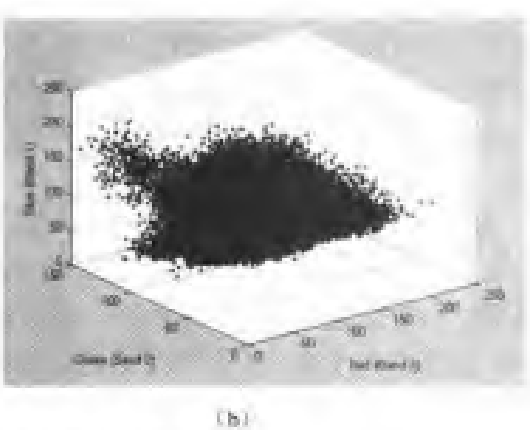
```
rA = A(:,:,1); gA = A(:,:,2); bA = A(:,:,3);
figure, plot3(rA(:),gA(:),bA(:),'.'); grid('on')
xlabel('Red (Band 3)'); ylabel('Green (Band 2)'); ...
zlabel('Blue (Band 1)')
rB = B(:,:,1); gB = B(:,:,2); bB = B(:,:,3);
```

```
figure, plot3(rB(:),gB(:),bB(:),'.'); grid('on')
xlabel('Red (Band 3)'); ylabel('Green (Band 2)'); ...
zlabel('Blue (Band 1)')
```

生成图 28-9。



(a)



(b)

图 28-9　去相关拉伸前后的颜色散点图

2．添加线性对比拉伸

现在进行同样的转换，但是在去相关拉伸以后进行线性对比拉伸。



```
imshow(A); C = decorrstretch(A,'Tol',0.01); figure; imshow(C)
```

结果如图 28-10（b）所示。



(a)



(b)

图 28-10　去相关拉伸后再进行线性对比拉伸前后的图像

添加线性对比拉伸以后，通过进一步拓宽颜色范围，改善了图像效果。在这种情况下，转换后的颜色范围在每个色带内映射到 0.01 和 0.09 之间的正态化区间。

28.2　去噪

数字图像中往往存在各种类型的噪声。产生噪声的途径可以有几种，与生成图像的方法有关。如：

- 如果图像是用照片扫描得到的，则胶卷上的灰尘是噪声源。胶卷损坏、扫描操作中

都可以引起噪声。

- 如果图像直接来源于数字设备，则获取数据的设备可以引起噪声。
- 图像数据的电子传输可以引起噪声。

工具箱提供了多个不同的方法来删除和减少图像中的噪声。不同方法对于不同类型的噪声具有更好的效果。可用的方法包括：

- 线性滤波；
- 中值滤波；
- 自适应滤波。

为了模拟上面列出的噪声效果，工具箱提供了 imnoise 函数，利用它可以在图像上添加不同类型的噪声。后面的例子会用到该函数。

28.2.1 线性滤波

可以用线性滤波来删除一定类型的噪声。某些滤波器，如均值滤波器或高斯滤波器比较适用。例如，均值滤波器对于从照片上删除灰尘噪声比较有用。因为每个像素的值设置为它邻域内像素值的均值，灰尘引起的局部变化也就减小了。

28.2.2 中值滤波

中值滤波与均值滤波的相似之处在于，每个像素的值由输入图像中对应像素邻域内的像素值确定。不同的是，均值滤波是根据输入图像中对应像素邻域内的像素值的均值确定输出图像中对应像素的值，而中值滤波是根据中值来确定的。中值对异常值的敏感性比均值的小，所以，中值滤波器可以在不减小图像对比度的情况下剔除这些异常值。medfilt2 函数实现了中值滤波。

下面的例子分别用均值滤波器和 medfilt2 函数删除图像中的食盐和胡椒粉状噪声。这种类型的噪声由设置为黑色或白色的随机像素点组成。在两种情况下，用于滤波的邻域大小都是 3×3。

（1）读入并显示图像。

```
I = imread('eight.tif');
imshow(I)
```

生成图 28-11。



图 28-11 载入图像

（2）添加噪声。

```
J = imnoise(I,'salt & pepper',0.02);
figure, imshow(J)
```

生成图 2-12。

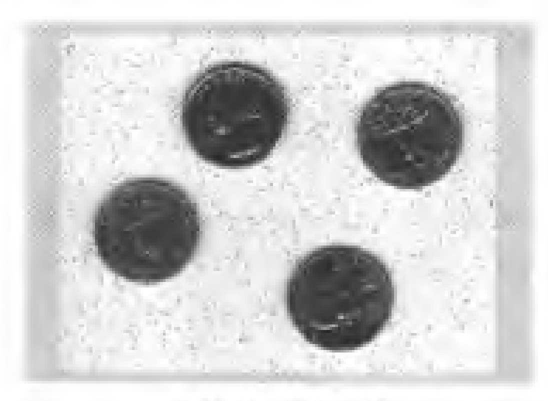


图 28-12 添加噪声

（3）用均值滤波器对有噪声的图像进行滤波，然后显示处理结果。

```
K = filter2(fspecial('average',3),J)/255;
figure, imshow(K)
```

生成图 28-13。

（4）现在用中值滤波器对有噪声的图像进行滤波并显示处理结果。

```
L = medfilt2(J,[3 3]);
figure, imshow(L)
```

生成图 28-14。比较图 28-13 和图 28-14 可以看出，后者的处理结果更好，图像中的对象边界更清晰。

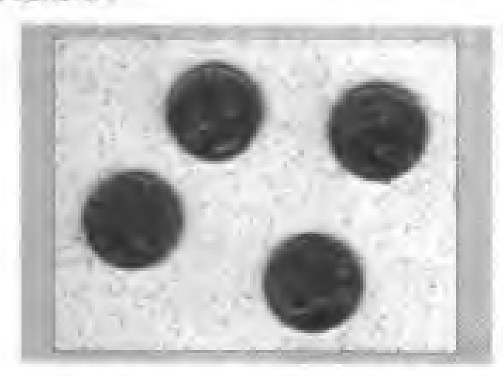


图 28-13 均值滤波后的效果



图 28-14 中值滤波效果

28.2.3 自适应滤波

用 wiener2 函数，采用 Wiener 滤波器（一种线性滤波器）根据图像的局部变异进行自适应滤波。变异大的地方，wiener2 函数进行比较小的平滑；变异小的地方，wiener2 函数进行能够比较大的平滑。

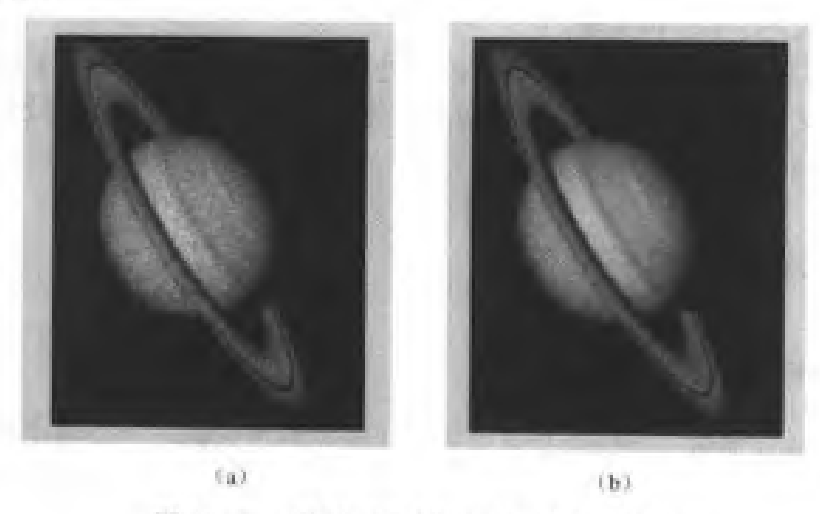
这种方法常常获得比线性滤波更好的效果。自适应滤波器比可以比较的线性滤波器更可取，它可以保留图像的边界和其他高频部分。另外，使用自适应滤波器没有设计任务，

wiener2 函数进行所有的前期计算并实现滤波器。但是，wiener2 函数确实要比线性滤波器花费更多时间。当噪声是常数幂值附加噪声如高斯噪声时，wiener2 函数工作得最好。

下面的例子将 wiener2 函数应用于一幅添加了高斯噪声的土星图像。

```
RGB = imread('saturn.png');
I = rgb2gray(RGB);
J = imnoise(I,'gaussian',0,0.005);
K = wiener2(J,[5 5]);
imshow(J)
figure, imshow(K)
```

生成图 28-15。



(a) (b)

图 28-15 对有高斯噪声的土星图像进行滤波

第 29 章　图 像 配 准

图像配准指的是将同一场景的两幅或多幅图像进行对准。一个典型的应用是，将一幅图像（称为基准图像）作为其他图像（称为输入图像）的参照进行比较。图像配准的目的是，通过对输入图像进行空间变换，使输入图像与基准图像对准。

空间变换将一幅图像中的位置映射到另一幅图像中的新位置。确定使图像对准的空间变换参数对于图像配准来说很关键。

图像配准常常用作其他图像处理应用的前处理步骤。例如，可以用图像配准对准地表卫星图像或核磁共振图像。配准以后，可以对图像进行比较，看河流如何迁移，大地如何泛滥，或者确定核磁共振图像上是否能看到肿瘤。

29.1　配准图像的一般过程

29.1.1　点映射

图像处理工具箱提供了一些支持点映射的工具，利用它们，可以确定使图像与其他图像配准的变换参数。进行点映射时，在成对图像中选择点来确认图像中的相同特征和标志。然后，根据这些控制点的位置来推导出某种空间映射关系。

使用点映射的图像配准包括以下步骤：

- 将图像读入到 MATLAB 工作空间；
- 指定图像中的成对控制点；
- 保存控制点对；
- 用反相关调整控制点（这一步可选）；
- 指定要使用的变换类型，并根据控制点对推测参数。
- 对没有配准的图像进行变换，使之对准。

29.1.2　示例：将数字航空照片配准成数字正色投影照片

下面的例子将覆盖同一区域的一幅数字航空照片配准成数字正色投影照片。航空照片在几何上是不正确的，其中包含了相机、透视、地形地貌建筑的影响和镜头等导致的失真，而且它对于地表没有作任何特殊的配准处理。

正色摄影照片经过正交校正，已经清除了相机、透视和地形地貌的影响。它还经过了地理配准（和地理编码）--数字正色摄影图像的列和行与地表平面坐标系统的坐标轴对准，每个像素中心对应于确定的地理图形位置，图像上的一个像素对应于地图上的 $1m^2$。

1．将图像读入 MATLAB

本例中，基准图像是图像 westconcordorthophoto.png，它是经过地理配准后的正色摄影

图像，它是全色（灰度）图像。要进行配准的图像是 westconcordareial.png，它是航空照片，RGB 图像。

```
orthophoto = imread('westconcordorthophoto.png');
figure, imshow(orthophoto)
unregistered = imread('westconcordaerial.png');
figure, imshow(unregistered)
```

生成图 29-1 和图 29-2，分别为要配准的输入图像和基准图像。不必将图像读入到 MATLAB 工作空间中。cpselect 函数接受灰度图像的文件规范。但是，如果想用反相关来调整控制点的位置，则图像必须位于工作空间中。



图 29-1 要配准的图像



图 29-2 基准图像

2．在图像中选择控制点

工具箱提供了一个称为控制点选择工具的交互工具，可以利用它在两幅图像中选择成对的对应控制点。控制点是两幅图像中都能找到的标记，比如道路的交叉口或者自然景物等。

在 MATLAB 提示符后面输入 cpselect 函数，将输入图像和基准图像作为变量，启动该工具。

注意，没有配准的图像是 RGB 图像。因为控制点选择工具只接受灰度图像，本例只传递彩色图像的一个层面给 cpselect 函数。

```
cpselect(unregistered(:,:,1),orthophoto)
```

cpselect 函数显示输入图像和基准图像的两个视图，用鼠标进行点击，可以选择控制点。如图 29-3 所示。

3．将控制点对保存到 MATLAB 工作空间

在控制点选择工具中，单击"File"菜单并选择"Save Points to Workspace"选项。例如，控制点选择工具在输入图像中返回下面的控制点系列。这些值表示空间坐标，左边一列是 x 坐标，右边一列是 y 坐标。

```
input_points =
  120.7086   93.9772
  319.2222   78.9202
  127.9838  291.6312
  352.0729  281.1445
```

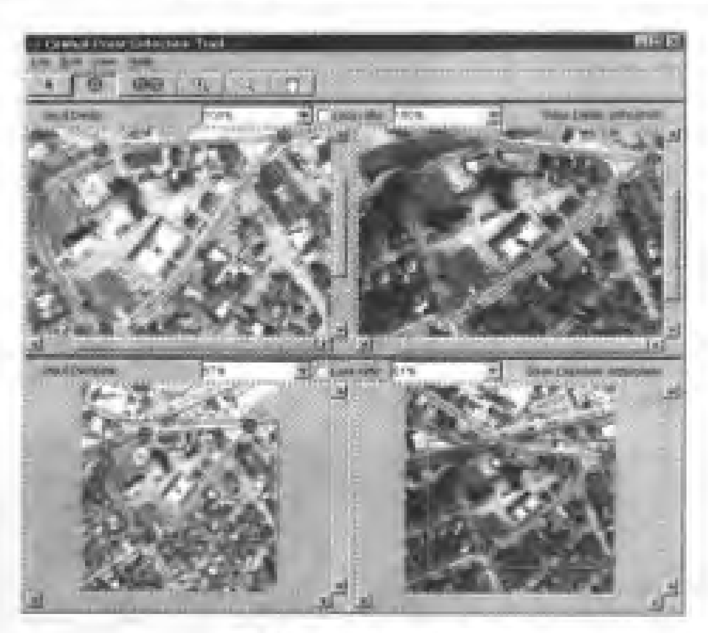


图 29-3　选择控制点

4．调整控制点对的位置

本步是可选步，用反相关来调整 cpselect 函数选择的控制点的位置。本步用到 cpcorr 函数。需要注意的是，该函数只能调整比例和方向相同的图像。

```
input_points_corr = cpcorr(input_points,base_points,...
unregistered(:,:,1),orthophoto)
input_points_corr =
  120.7086    93.9772
  319.2222    78.9202
  127.1046   289.8935
  352.0729   281.1445
```

5．指定变换类型计算参数

这一步将控制点传递给 cp2tform 函数，确定变换参数。cp2tform 函数是一个数据拟合函数，确定基于控制点几何关系的变换。cp2tform 函数将参数返回到名为 TFORM 的几何变换结构中。

使用 cp2tform 函数时，必须指定要进行的变换类型。cp2tform 函数可以计算 5 种类型的变换。必须选择用哪种变换纠正输入图像中存在的失真现象。

引起航空照片失真的最主要原因是照相机的透视成像方式。本地区比较平坦，所以不考虑地形地貌的影响，图像配准通过使用投影变换可以纠正它。投影变换还将图像旋转到与基准数字正色投影图像下面的映射坐标系统对准。

```
mytform = cp2tform(input_points,base_points,'projective');
```

6．变换没有配准的图像

最后一步，变换输入图像，使之与基准图像对准。将输入图像和定义变换的 TFORM 结构传给 imtransform 函数，进行变换。imtransform 函数返回变换后的图像。

```
registered = imtransform(unregistered,mytform)
```

配准结果如图 29-4 所示。比较变换后的图像和图 29-2 所示的基准图像，观察配准效果。



图 29-4　配准后的图像

29.2　支持的变换类型

cp2tform 函数可以计算 6 种类型的变换。表 29-1 按照复杂度列出了这些变换，还列出了每一种失真类型。

前 4 种变换，即“linear conformal”、“affine”、“projective”和“polynomial”为全局变换。在这些变换中，单一的数学表达式适用于整幅图像。最后两种变换，即“piecewise linear”和“lwm”（局部加权均值）都是局部变换。在这些变换中，不同的数学表达式适用于图像中不同的区域。

表 29-1　变换类型及其描述

变换类型	描　述	最小控制点对数	示　例
'linear conformal'	当输入图像中的形状没有改变，但是图像经过平移、旋转和比例等的组合变换以后发生失真时使用本变换。变换以后，直线仍然是直线，平行线仍然是平行线	2 对	
'affine'	当输入图像中的形状出现错切现象时使用本变换。变换以后，直线仍然是直线，平行线仍然是平行线，但是矩形变成了平行四边形	3 对	
'projective'	景物显得倾斜时使用本变换。变换后直线仍然是直线，但是平行线不再平行	4 对	
'polynomial'	图像中的对象发生弯曲时使用本变换。多项式的阶数越高，拟合效果越好，但是生成的图像比基准图象包含更多的曲线	6 对(2 阶) 10 对(3 阶) 16 对(4 阶)	
'piecewise linear'	当图像中的变形现象具有分段性时使用本变换	4 对	
'lwm'	当变形有局部性的变化，而且分段线性的条件不够充分时使用本变换	6 对(推荐 12 对)	

29.3　选择控制点

在要配准的成对图像中指定控制点，需要使用控制点选择工具 cpselect。该工具在基准图像后面显示要配准的图像，即输入图像。按下面 4 步指定控制点。

（1）启动工具，指定输入图像和基准图像。

（2）查看图像，寻找在两幅图像中都可辨认的可视元素。cpselect 函数提供了许多方法来查看图像，包括平移、缩放等。

（3）指定两幅图像中的匹配控制点。

（4）将控制点保存到 MATLAB 工作空间中。

图 29-5 显示了第 1 次启动工具时的默认外观。

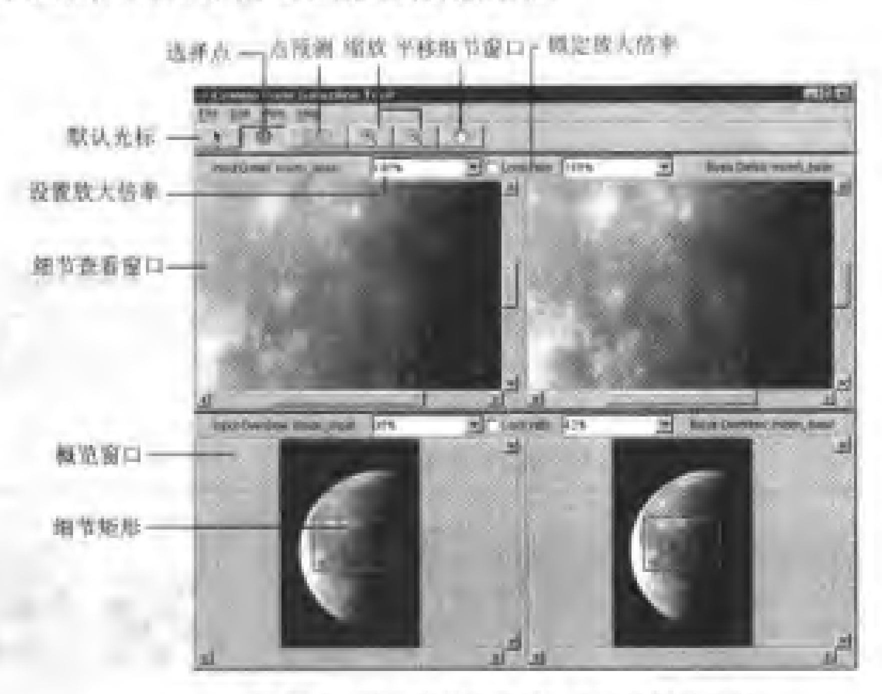


图 29-5　第 1 次启动选择工具时的默认外观

1．启动控制点选择工具

要使用控制点选择工具，在 MATLAB 提示符后键入 cpselect 命令就可以了。作为变量，指定输入图像和基准图像。

为了进行演示，下面的代码片段将一幅图像读入 MATLAB 工作空间的变量 moon_base 中。然后将它进行大小变形，保存到另一个变量 moon_input 中，这就是需要进行配准，删除大小变形的图像。下面的代码启动 cpselect 工具，指定这两幅图像。

```
moon_base = imread('moon.tif');
moon_input = imresize(moon_base, 1.2);
cpselect(moon_input, moon_base);
```

控制点选择工具启动时，它包含 4 个图像显示窗口。上面的两个窗口称为细节窗口，显示当前正在操作的图像部分的视图。输入图像在左侧，基准图像在右侧。下面两个窗口称为

概览窗口，这两个窗口显示完整的图像。输入概览图像位于左侧，基准概览图像位于右侧。

概览窗口中的图像上有一个矩形，称为细节矩形。该矩形定义细节窗口中要显示的图像部分。默认时，启动后细节矩形覆盖整幅图像的四分之一，位置在图像中心。

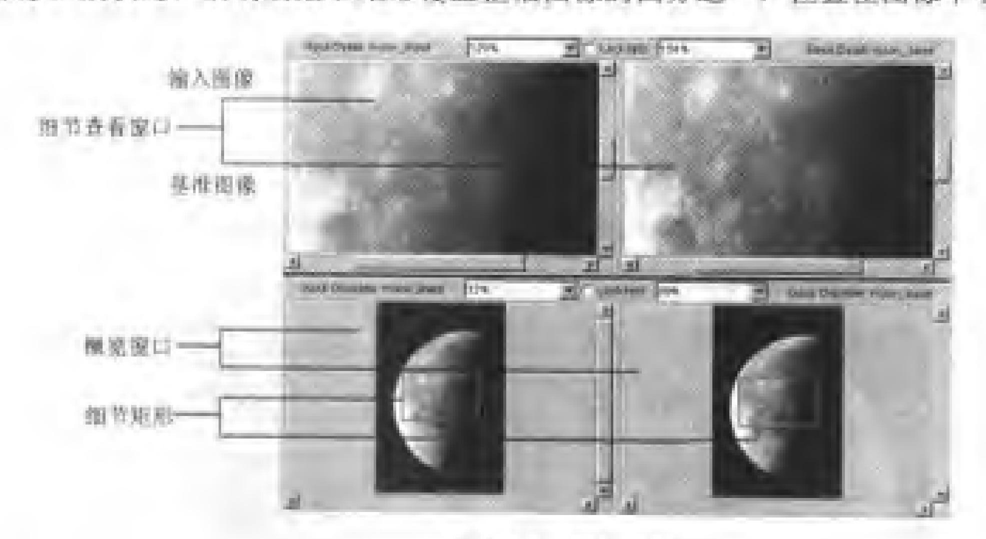


图 29-6　默认视图

2. 查看图像

默认时，cpselect 函数在概览窗口中显示整幅基准图像和输入图像，并且在细节窗口中显示这些图像的一部分。但是，要找到两幅图像中都可见到的可视元素，可能需要移动或放大图像。下面介绍了改变图像视图的一些方法：

- 用滚动条查看图像的其他部分；
- 用细节矩形改变视图；
- 移动细节窗口中显示的图像；
- 缩放图像；
- 指定图像的放大倍率；
- 锁定输入图像和基准图像的相对放大倍率。

3. 指定匹配控制点对

控制点选择工具的主要功能就是要在基准图像和输入图像上选择控制点。启动控制点选择工具以后，通过在输入图像和基准图像上点击来指定控制点。输入图像中指定的每个点必须与基准图像中的匹配。下面介绍选择控制点对的方法。

- 手工选择控制点对；
- 使用控制点预测。

这里还介绍如何在创建控制点以后移动和删除它们。

（1）手工选择控制点

按照以下步骤，在图像上指定一对控制点：

① 单击控制点选择按钮，默认时控制点选择模式是激活的。

② 将光标放在图像中选定的地物上。光标变成形状。可以在细节窗口或概览窗口中选择控制点。

③ 单击鼠标键，cpselect 函数在指定的位置上放一个控制点标记。细节窗口和概览窗

口中都会显示这个标记。

④ 为了创建这个控制点的匹配点，在对应的细节窗口或概览窗口中移动光标。例如，如果开始选择的控制点在输入窗口中，则将光标移动到基准窗口中。

⑤ 单击鼠标键，cpselect 函数将控制点标记放在指定的位置上。因为控制点完成了配对，标记的外观会显示激活的匹配状态。注意，第 3 步中选择的第 1 个控制点的外观也改变成激活的匹配状态。

根据不同的匹配状态，会有不同的标记显示，如图 29-7 中所示。

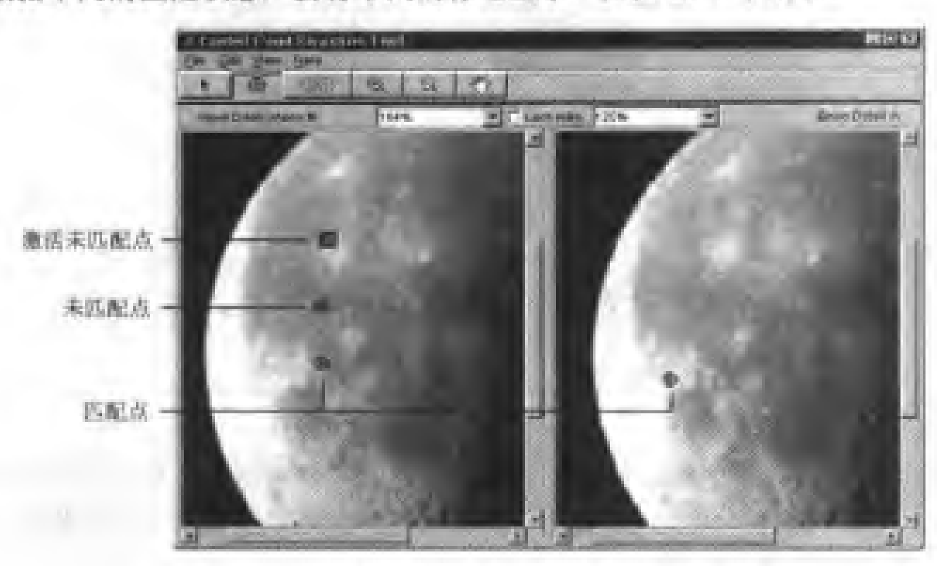


图 29-7　不同匹配状态使用不同的标记

（2）使用控制点预测

在对应的细节窗口和概览窗口中移动光标，可以选择匹配控制点。如果不采用这种选择方式，可以让控制点选择工具自动估计指定控制点之间的匹配关系。控制点选择工具根据先前选择的控制点之间的几何关系来确定匹配控制点的位置。

为了演示控制点预测，图 29-8 在显示了输入图像中选择的 4 个控制点，这 4 个点组成了一个方形的 4 个角。左图中显示了第 4 个点的选择，对应的预测点显示在右边窗口中。注意控制点选择工具是如何将预测点放在相对于其他控制点的同一位置并组成方形的右下角的。

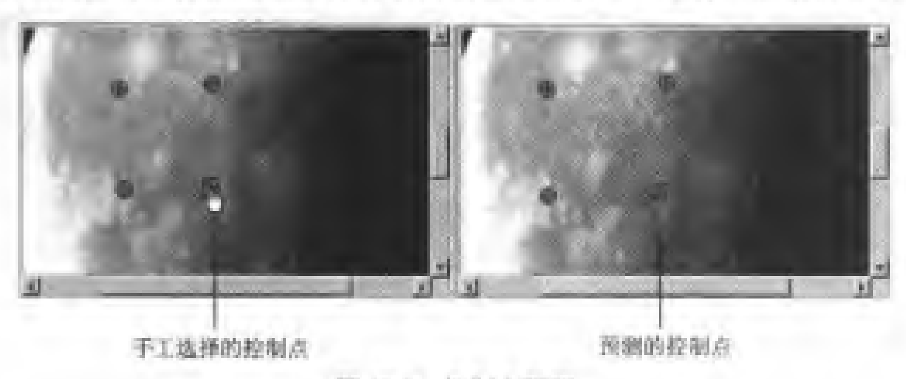


图 29-8　控制点预测

按照以下步骤使用控制点预测：

① 单击控制点预测按钮。

② 将鼠标光标移动到任何图像的任何位置，光标变成 形状。

③ 单击鼠标键，控制点选择工具将一个控制标记放在指定的位置上，并且将匹配控制标记放在所有其他窗口中。预测点的标记包含字母"P"，表示它是预测控制点。图 29-9 显示了未匹配、匹配和预测状态下的预测点。

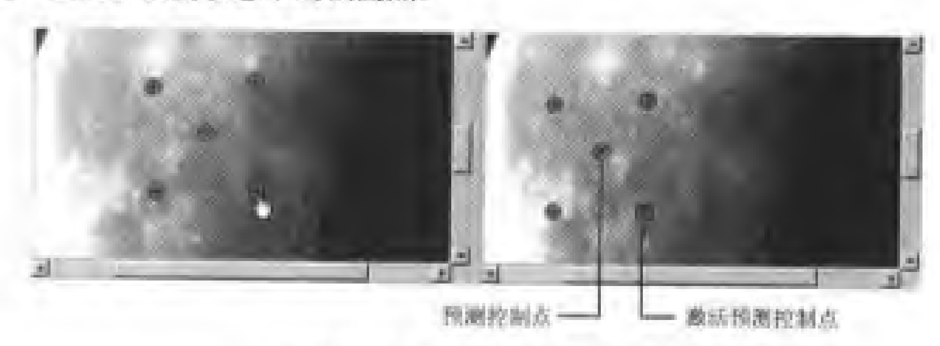


图 29-9　几种状态下的预测控制点

（3）控制点状态

控制点标记的外观表明了它们的当前状态。第 1 次选择控制点时，它的状态是激活和未匹配的。为控制点选择匹配点时，两个标记的外观都变成匹配状态。

表 29-2 用标记列出了所有可能的控制点状态。cpselect 函数在单独的名为 Legend（图例）的窗口中显示这个列表。默认时该图例是可见的，但是可以用"View"菜单中的"Legend"选项控制它的可见性。

表 29-2　控制点状态

标记	状　态	描　述
	激活，未匹配	当前，点被选择，但是没有匹配点。这是大部分点的初始状态
	激活，匹配	点被选择并且有匹配点
	激活，预测	点为预测点，如果移动它的位置，则点改变到激活匹配状态
	未匹配	点未被选择并且未匹配，必须在能够创建它的匹配点以前选择它
	匹配	点有一个匹配点
	预测	点在点预测过程中被 cpselect 函数所添加

（4）移动控制点

按照以下操作移动控制点：

① 单击控制点选择按钮 或默认的光标按钮 。

② 将光标放在要移动的控制点上方。

③ 按下并拖拉鼠标，移动时控制点变为激活状态。

如果移动预测后的控制点，控制点的状态变成矩形控制点。

（5）删除控制点

按照以下步骤删除控制点：

① 单击控制点选择按钮 或默认的光标按钮 。

② 单击要删除的控制点。它的状态改变为激活的。如果控制点有一个匹配点，则两个点都变成激活的。

③ 用回退键、Delete 键或者其他方法删除该点。

4．保存控制点

指定控制点对以后，必须将它们保存到 MATLAB 工作空间中，以便在后面的图像配准中使用它们。按照以下步骤保存控制点：

（1）在控制点选择工具中选择“File”菜单。

（2）选择“Save Points to Workspace”选项。控制点选择工具显示图 29-10 所示的对话框。

默认时，控制点选择工具保存控制点的 x 坐标和 y 坐标，这里用两个数组 input_points 和 base_points 保存。这两个数组是 $n\times2$ 的数组，其中，n 是所选择的合法控制点的个数。例如，如果选择了 4 对控制点，下面是 input_points 数组的一个示例。左列的值表示 x 坐标，右列的值表示 y 坐标。

```
input_points =
   215.6667   262.3333
   225.7778   311.3333
   156.5556   340.1111
   270.8889   368.8889
```

不管什么时候退出控制点选择工具，都会询问是否保存控制点。

在“Save Points to Workspace”对话框中选择“Structure with all points”核选框，可以保存控制点选择工具的当前状态，如图 29-11 所示。



图 29-10 “Save Points to Workspace”对话框

图 29-11 “Save Points to Workspace”对话框

该选项将指定的所有控制点的位置和它们的当前状态保存在 cpstruct 结构中。

```
cpstruct =
           inputPoints: [4x2 double]
            basePoints: [4x2 double]
        inputBasePairs: [4x2 double]
                   ids: [4x1 double]
          inputIdPairs: [4x2 double]
           baseIdPairs: [4x2 double]
      isInputPredicted: [4x1 double]
       isBasePredicted: [4x1 double]
```

这个选项在某些情况下很有用，比如花了很长时间选择了很多点，并且希望在重新开始工作时保留未匹配的点和预测点的情况。控制点选择工具不将未匹配点和预测点包括在 input_points 和 base_points 数组中。

第 30 章 图像恢复

30.1 理解图像恢复

30.1.1 影响图像质量的原因

下面介绍一些图像恢复的背景，包括进行图像恢复的原因和图像恢复模型两个方面的内容。

影响图像质量的因素主要有下面一些：

- 图像捕获过程中镜头发生了移动，或者暴光时间过长；
- 场景位于焦距以外、使用了广角镜、大气干扰或短时间的暴光导致捕获到的光子减少；
- 共焦显微镜中出现散光变形。

30.1.2 图像恢复模型

一幅质量改进或退化的图像可以近似地用方程 $g = Hf + n$ 表示，其中 g 为图像，H 为变形算子，又称为点扩散函数（PSF），f 为原始的真实图像，n 为附加噪声，它在图像捕获过程中产生并且使图像质量变坏。

上面的模型中，PSF 是一个很重要的因素，它的值直接影响到恢复后图像的质量。下面结合一个实例来进行演示。

实例给出一幅清晰的图像，然后用 PSF 进行卷积操作，把图像故意弄模糊。其中用到了 fspecial 函数，它创建一个模拟移动模糊化的 PSF，同时指定了模糊的长度和角度，长度单位为像素，角度单位为度。一旦创建了 PSF，实例就可以用 imfilter 函数将 PSF 用于原始图像 I，生成模糊化后的图像 Blurred。

```
I = imread('peppers.png');
I = I(60+[1:256],222+[1:256],:); % crop the image
figure; imshow(I);

LEN = 31;
THETA = 11;
PSF = fspecial('motion',LEN,THETA); % create PSF
Blurred = imfilter(I,PSF,'circular','conv');
figure; imshow(Blurred);
```

生成图 30-1。


(a)


(b)

图 30-1 原图像和模糊化后的图像

30.2 用函数恢复图像

工具箱中有4个图像恢复函数，如表30-1所示。

表 30-1 图像恢复函数

deconvwnr	用 Wiener 滤波器实现图像恢复
deconvreg	用 regularized 滤波器实现图像恢复
deconvlucy	用 Lucy-Richardson 算法实现恢复
deconvblind	用 blind 去卷积算法实现图像恢复

这4个函数都将PSF和模糊图像作为主要变量。deconvwnr函数求取最小二乘解，deconvreg函数求取有约束的最小二乘解，可以设置对输出图像的约束。使用这些函数中的任何一种，都应该提供一些与噪声相关的信息来减少恢复过程中可能出现的噪声扩大。

deconvlucy函数实现了一个加速衰减的Lucy-Richardson算法。本函数采用优化技术和泊松统计量进行多次迭代。使用该函数，不需要提供有关模糊图像中附加噪声的信息。

deconvblind函数使用的是盲去卷积算法，它在不知道PSF的情况下进行恢复。调用deconvblind函数时，将PSF的初值作为一个变量进行传递。该函数除了返回一个修复后的图像以外，还返回一个修复后的PSF。这里使用与deconvlucy函数相同的衰减和迭代模型。

30.2.1 用 Wiener 滤波器进行恢复

用deconvwnr函数，采用Wiener滤波器恢复图像。在图像的频率特征和附加噪声已知的情况下，采用Wiener去卷积比较有效。

本例使用“图像恢复模型”一小节中创建的模糊图像，指定同一个PSF函数。本例演示了导致模糊化的函数PSF的重要性。得到准确的PSF时，恢复的结果会比较好。

（1）把图像读入MATLAB工作空间（为了加速恢复操作，示例还对图像进行裁剪）。

```
I = imread('peppers.png');
I = I(10+[1:256],222+[1:256],:);
figure;imshow(I);
```

（2）创建 PSF

```
LEN = 31;
THETA = 11;
PSF = fspecial('motion',LEN,THETA);
```

（3）将图像模糊化

```
Blurred = imfilter(I,PSF,'circular','conv');
figure; imshow(Blurred);
```

（4）恢复图像

```
wnr1 = deconvwnr(Blurred,PSF);
figure;imshow(wnr1);
```

生成图 30-2。

通过提供 deconvwnr 函数支持的优化变量值，可以影响去卷积结果。使用这些变量，可以指定噪声-信号幂值和/或提供自相关函数来帮助改善恢复的结果。

30.2.2　用 regularized 滤波器进行恢复

采用 deconvreg 函数，用 regularized 滤波器恢复图像。当知道附加噪声的部分信息时，使用 regularized 滤波器比较有效。

下面的例子首先对给定的图像进行模糊化，然后用高斯滤波器进行恢复。图像中的附加噪声通过添加方差为 V 的高斯噪声到模糊图像中进行模拟。

（1）将一幅图像读入 MATLAB 工作空间。本例使用裁剪来减小要恢复的图像的大小。

```
I = imread('tissue.png');
I = I(125+[1:256],1:256,:);
figure;imshow(I);
```

生成图 30-3。

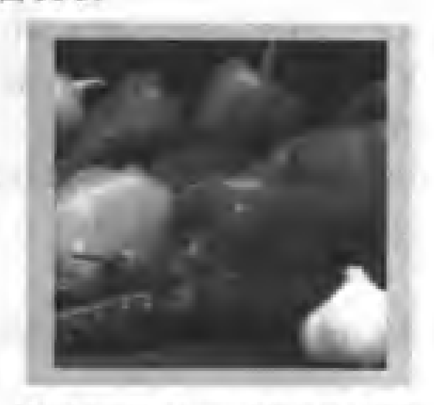


图 30-2　用 Wiener 滤波器恢复图像

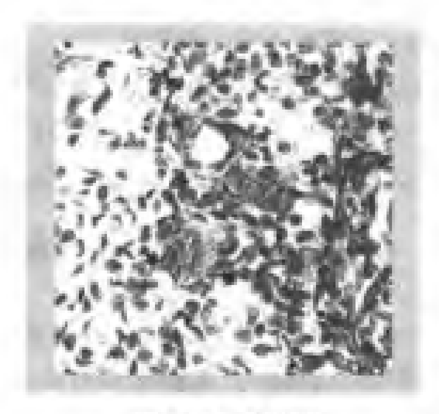


图 30-3　原图像

（2）创建 PSF。

```
PSF = fspecial('gaussian',11,5);
```

（3）模糊化图像并添加噪声。

```
Blurred = imfilter(I,PSF,'conv');
V = .02;
```

```
BlurredNoisy = imnoise(Blurred,'gaussian',0,V);
figure;imshow(BlurredNoisy);
```

生成模糊化后的图像，如图30-4所示。

（4）用deconvreg函数恢复图像，指定PSF和噪声幂次NP。

```
NP = V*prod(size(I));
[reg1 LAGRA] = deconvreg(BlurredNoisy,PSF,NP);
figure,imshow(reg1);
```

恢复后的图像如图30-5所示。

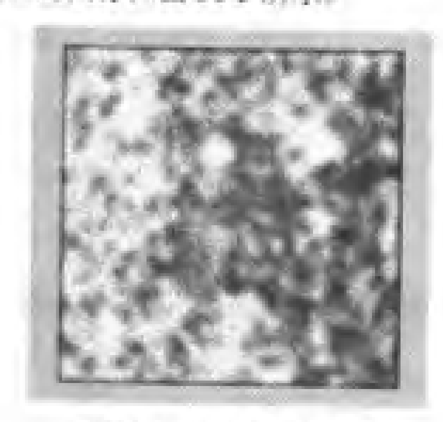


图30-4 模糊化后的图像

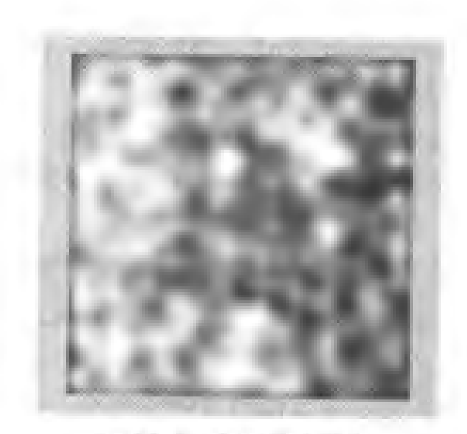


图30-5 恢复后的图像

30.2.3 用Lucy-Richardson算法进行恢复

使用deconvlucy函数，用加速衰减Lucy-Richardson算法恢复图像。假定泊松噪声统计量，用PSF进行卷积时，该算法使生成的图像是模糊图像实例的可能性最大。已知PSF，但对图像中的附加噪声知之甚少时，deconvlucy函数比较有效。该函数实现了原始Lucy-Richardson最大似然算法的几个改写版本，用它们可以完成复杂的图像恢复任务。使用这些改写版本，可以：

- 减小图像恢复中的噪声扩大效应；
- 解释为什么图像质量不均匀；
- 控制图像在焦距外和背景噪声问题；
- 通过二次采样来改进恢复图像的质量。

1．减小噪声扩大效应

噪声扩大是最大似然法面临的一个共同问题，它试图尽可能接近地拟合数据。经过多次迭代以后，恢复的图像会有一些斑点，在比较低的信号-噪声比率上观察平滑物体时尤其如此。这些斑点不代表图像中的任何真实结构，而是人为造成的。

为了控制噪声扩大问题，deconvlucy函数使用了一个衰减参数DAMPAR。该参数指定最后生成的图像与原始图像之间偏差的阈值水平。对于偏离发生在原值附近的像素，迭代终止。

2．解释非均匀的图像质量

图像恢复中存在的另一个复杂性在于数据中可能包括坏的像素点，或者，接收到的像

素的质量可能随时间和位置发生变化。通过给 deconvlucy 函数指定 WEIGHT 数组参数，可以指定忽略图像中的某些像素。忽略像素，只需要将 WEIGHT 数组中对应于图像中像素的元素权重设置为 0 就可以了。

该算法主要根据坏像素周围像素的信息来预测坏像素的值。

3．控制相机的 Read-Out 噪声

电荷偶合装备探测器主要有两个方面的噪声：

- 泊松分布的光子计数噪声；
- 高斯分布的 Read-Out 噪声。

Lucy-Richardson 迭代主要解释第 1 种噪声。还必须解释第 2 种噪声，因为它会导致具有低水平附带光子的像素产生负值。

deconvlucy 函数用 READOUT 输入参数控制相机 Read-Out 噪声。该参数的值是 Read-Out 噪声方差和背景噪声的和。READOUT 参数的值指定一个偏移值，确保所有值都是正的。

4．示例：用 deconvlucy 函数恢复图像

本例首先用高斯滤波器 PSF 卷积来获得一幅模糊有噪声的图像，然后向模糊图像中添加方差为 V 的高斯噪声。

（1）将图像读入 MATLAB 工作空间。

```
I = imread('board.tif');
I = I(50+[1:256],2+[1:256],:);
figure;imshow(I);
```

显示图 30-6。

（2）创建 PSF。

```
PSF = fspecial('gaussian',5,5);
```

（3）模糊化图像并添加噪声。

```
Blurred = imfilter(I,PSF,'symmetric','conv');
V = .002;
BlurredNoisy = imnoise(Blurred,'gaussian',0,V);
figure;imshow(BlurredNoisy);
```

得到模糊化后的图像，如图 30-7 所示。

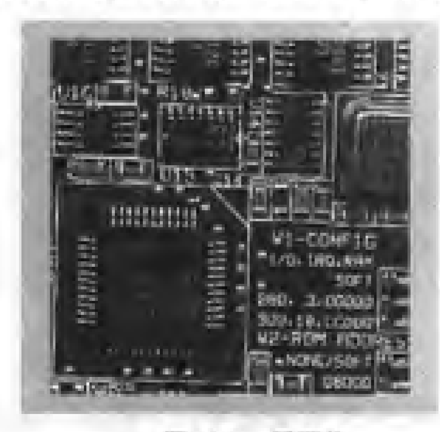


图 30-6　原图像



图 30-7　模糊化后的图像

指定与模糊化时相同的 PSF，将最高迭代次数设置为 5，用 deconvlucy 函数恢复模糊有噪声的图像。

```
luc1 = deconvlucy(BlurredNoisy,PSF,5);
figure; imshow(luc1);
```

恢复的图像如图 30-8 所示。

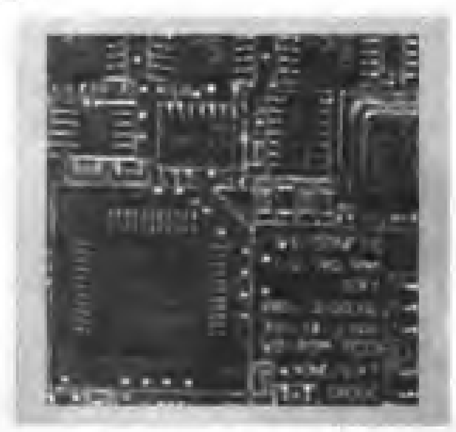


图 30-8　恢复后的图像

30.2.4　用盲去卷积算法进行恢复

使用 deconvblind 函数，采用盲去卷积算法恢复图像。该算法用在不知道失真信息的情况下比较有效。deconvblind 函数使用与加速衰减 Lucy-Richardson 算法近似的迭代处理来同时恢复图像和 PSF。

与 deconvlucy 函数类似，deconvblind 函数也实现了几个原 Lucy-Richardson 最大似然算法的改写版本来完成复杂的图像恢复任务。

下面的例子创建一个模拟的模糊图像，然后用 deconvblind 函数进行恢复。示例采用不同的参数选项进行了恢复操作。

（1）将图像数据读入 MATLAB 工作空间。

```
I = imread('cameraman.tif');
figure; imshow(I);
```

生成图 30-9。



图 30-9　原图像

（2）创建 PSF。

```
PSF = fspecial('motion',13,45);
figure; imshow(PSF,[],'notruesize');
```

生成原图像的 PSF 图，如图 30-10 所示。

（3）模糊化图像。

```
Blurred = imfilter(I,PSF,'circ','conv');
figure; imshow(Blurred);
```

模糊化后的图像如图 30-11 所示。



图 30-10　原图像的 PSF 图



图 30-11　模糊化后的图像

（4）恢复图像，设置 PSF 大小的初值。

为了确定 PSF 的大小，检查模糊图像并测量强度明显很高的区域周围的模糊像素。示例模糊图像中，可以测量人衣袖轮廓附近的模糊区域。因为 PSF 的大小比它包含的值更重要，所以可以指定一个元素都是 1 的数组，作为初始的 PSF。

```
INITPSF = ones(size(PSF));
[J P]= deconvblind(Blurred,INITPSF,30);
figure; imshow(J);
figure; imshow(P,[],'notruesize');
```

恢复的图像及其 PSF 图如图 30-12 和图 30-13 所示。



图 30-12　恢复后的图像

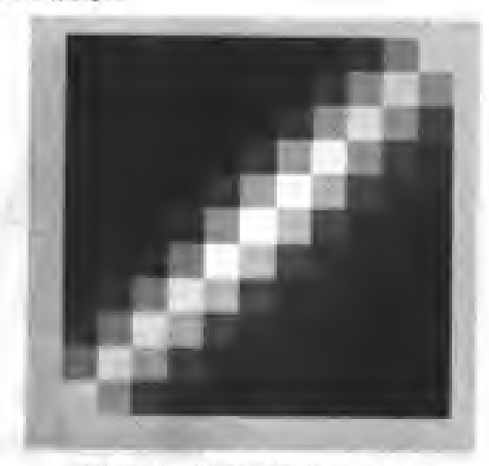


图 30-13　恢复后图像的 PSF 图

尽管 deconvblind 函数可以将图像恢复到很高的程度，但是明暗对照强烈区域周围的情况仍不能让人满意。下面的步骤重复恢复过程，试图通过剔除高对照区域和指定更好的 PSF 来达到更好的效果。

（5）创建 WEIGHT 数组，从恢复操作中排除高对照区域。

要想从处理过程中排除像素，需要创建一个与原始图像相同大小的数组，并将数组中与原始图像中要排除的像素对应的像素值设置为 0。为了创建 WIGHT 数组，本例组合使用了边缘探察和数学形态学处理来找出图像中的高对照区域。因为图像中的污点是线形的，示例对图像进行了两次膨胀。为了从处理过程中排除图像边缘像素（高对照区域），示例使用 padarray 函数将所有边缘像素的值指定为 0。

```
WEIGHT = edge(I,'sobel',.28);
se1 = strel('disk',1);
se2 = strel('line',13,45);
WEIGHT = ~imdilate(WEIGHT,[se1 se2]);
WEIGHT = padarray(WEIGHT(2:end-3,2:end-3),[2 2]);
figure; imshow(WEIGHT);
```

生成图 30-14。



图 30-14 WEIGHT 数组的图像

（6）改进 PSF 的初值。重新得到的 PSF p 显示了清晰的线性特征，它由去卷积的第 1 次传递返回。第 2 次传递使用了一个新的 PSF p1，它与 p 大小相同，但是振幅更小的像素值设置为 0。

```
P1 = P;
P1(find(P1 < 0.01))=0;
```

（7）指定 WEIGHT 数组和修改后的 PSF。

```
W=double(WEIGHT);
[J2 P2] = deconvblind(Blurred,P1,50,[],W);
figure; imshow(J2);
figure; imshow(P2,[],'notruesize');
```

生成恢复的图像及其 PSF 图，如图 30-15 和图 30-16 所示。



图 30-15　恢复后的图像



图 30-16　恢复后图像的 PSF 图

30.3　避免在恢复后的图像中出现 ringing 效应

恢复函数用到的离散傅里叶变换（DFT）假设图像的频率模式是周期性的。该假设在图像边缘造成了高频坠落现象。图 30-17 中，阴影区域表示实际的图像宽度；没有阴影的区域表示假设周期。

在恢复后的图像中，高频坠落会产生所谓的边缘相关 ringing 效应。图 30-18 中，注意图像中的水平和垂直图案。

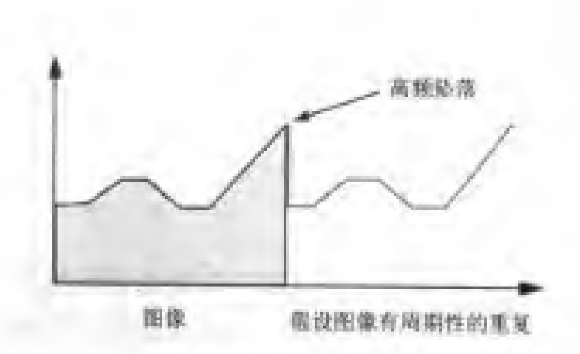


图 30-17　高频坠落现象

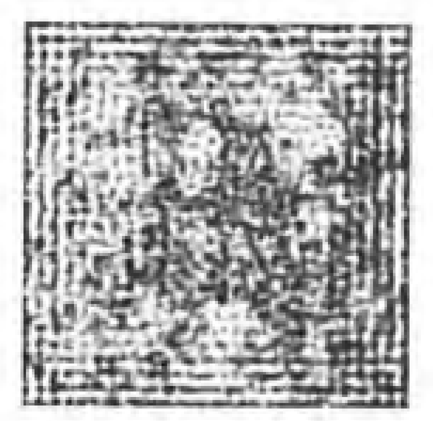


图 30-18　边缘相关 ringing 效应

为了避免出现 ringing 效应，将图像传递给恢复函数以前用 edgetaper 函数进行处理。该函数通过模糊化整幅图像来删除图像边缘的高频坠落现象，然后用原图像替换模糊图像的中心像素。此时，图像边缘逐渐变到一个更低的频率。

第 31 章　虚拟现实工具箱简介

虚拟现实工具箱是在三维虚拟现实环境下采用动态系统进行视图和交互的一种解决方案。它扩展了 MATLAB 和 Simulink 在虚拟现实图形学方面的能力，内容包括：

· 虚拟世界　用标准的虚拟现实模型语言（VRML）创建虚拟世界或三维场景。

动态系统　用 MATLAB 和 Simulink 创建和定义动态系统。

动画　查看 Simulink 信号驱动的移动的三维场景。

交互操作　可以改变虚拟世界中对象的位置和属性，或者在进行模拟时改变 Simulink 模型的参数。

为了提供一个完整的工作环境，虚拟现实工具箱还包括了其他一些组件：

VRML 查看器　使用虚拟现实工具箱查看器，或者对于 PC 平台，通过在 Web 浏览器上使用插件来显示虚拟世界。

VRML 编辑器　对于 PC 平台，使用 V-Realm 编辑器创建和编辑 VRML 代码。

对于 UNIX 或 Linux 平台，使用 MATLAB 文本编辑器来撰写 VRML 代码。

31.1　虚拟现实工具箱的特点

虚拟现实工具箱包括了许多用于创建和可视化动态系统的特点。它还用动态模型提供了即时动态交互功能。概括地讲，虚拟现实工具箱主要有下面一些特点：

VRML 支持　使用 VRML 定义虚拟世界。

MATLAB 接口　通过 MATLAB 接口控制虚拟世界。

Simulink 接口　使用虚拟现实工具箱提供的模块来连接 Sumulink 模型和虚拟世界。

VRML 查看器　用虚拟现实工具箱或 Web 浏览器查看虚拟世界。

VRML 编辑器　用 VRML 生成工具或文本编辑器创建虚拟世界。

实时工作室支持　支持用实时工作室创建的代码进行模拟。

硬件支持　含有使用特殊硬件设备的函数。

客户-服务器体系　对于单机或网络操作，提供了客户-服务器体系结构。

31.2　VRML 支持

VRML 是一种开放的、基于文本并且采用面向 WWW 的格式的 ISO 标准。使用 VRML 语言定义虚拟世界，这个虚拟世界就可以用 VRML 查看器进行显示，并且可以与 Simulink 模型相连接。

虚拟现实工具箱使用了当前 VRML97 规范中定义的许多高级特性。这里所讲的 VRML，始终指的是 VRML97 标准 ISO/IEC 14772-1：1997 中定义的 VRML。这种格式包

含了对三维场景、声音、内部操作和 WWW 定位等方面的描述。

虚拟现实工具箱分析虚拟世界的结构，确定可以获取哪些信号，并且使这些信号可以从 MATLAB 和 Simulink 中获取。

虚拟现实工具箱查看器还支持 VRML97 标准节点中的大部分，使得可以通过相关的虚拟世界完成控制。虚拟现实工具箱确保虚拟世界中所作的任何改变都在 MATLAB 和 Simulink 中有所反映。如果改变在虚拟世界中的视点，在 MATLAB 和 Simulink 中，对应的 vrworld 对象属性将发生改变。虚拟现实工具箱中包括了获取和改变虚拟世界属性的函数。

注意，因为有些 VRML 世界自动生成为 VRML1.0 版本，而虚拟现实工具箱不支持 VRML1.0，所以需要将这些世界保存为 VRML97 版本。如果使用的是 PC 平台，可以在 V-Realm Builder 中打开它们然后进行另存。利用其他一些商业软件也可以进行 VRML 1.0 至 VRML97 的转换。

31.3 MATLAB 接口

虚拟现实工具箱提供了与虚拟现实世界的灵活的 MATLAB 接口。创建 MATLAB 对象并将它们与虚拟世界连接以后，可以通过使用函数和方法来控制虚拟世界。

从 MATLAB 那里，可以设置 VRML 对象的位置和属性，创建图形用户界面的回调，将数据映射到虚拟对象。还可以用 VRML 查看器查看虚拟世界，确定它的结构并给所有可以获取的节点和它们的字段赋新值。

虚拟现实工具箱中包括有获取和改变虚拟世界属性以及保存 VRML 文件的函数。这些文件与虚拟世界的实际结构相对应。MATLAB 提供了用 MATLAB 对象对虚拟现实对象进行控制和操作的通信机制。

31.4 Simulink 接口

使用 Simulink 接口，可以实现对三维可视模型在一定时间段内的动态系统模拟。虚拟现实工具箱提供了直接连接 Simulink 信号和虚拟世界的模块。通过这个连接，可以将模型可视化为一个三维动画。

使用 Simulink 模块，可以实现大部分虚拟现实工具箱特性。一旦在 Simulink 图中包括了这些模块，就可以选择一个虚拟世界并使它与 Simulink 信号相连接。虚拟现实工具箱自动对虚拟世界进行扫描，以获取 Simulink 可以驱动的 VRML 节点。

所有 VRML 节点属性都罗列在层次树形查看器中。可以在该查看器中选择树的层次级别来进行控制。关闭“Block Parameters”对话框以后，Simulink 用与虚拟世界中选定节点相对应的输入和输出来进行更新。将这些输入与合适的 Simulink 信号相连接以后，就可以用 VRML 查看器查看整个模拟过程。

Simulink 通过虚拟现实工具箱模块提供了控制和操作虚拟现实对象的通信机制。

31.5　VRML 查看器

虚拟现实工具箱中包含一个查看器，它是查看虚拟世界的默认工具。第 33 章详细介绍该查看器的使用。

31.6　VRML 编辑器

虚拟现实工具箱中带了一个经典的 VRML 生成工具—V-Realm Builder。除了提供这个工具外，虚拟现实工具箱还提供了进行三维可视模拟所需要的完整的生成、开发和操作环境。

第 32 章 VRML 与 V-Realm 编辑器

本章介绍虚拟现实模型语言（VRML）的基本知识和可以生成和编辑 VRML 虚拟场景的编辑器，重点介绍 V-Realm 编辑器。

32.1 VRML 语言

VRML 是一种语言，使用它，可以在 VRML 查看器中显示三维对象。本节包括的内容有：

- VRML 的历史 VRML97 标准出台大事记。
- VRML 的坐标系统 VRML 坐标系统与 MATLAB 坐标系统有所不同。
- VRML 的文件格式 VRML 文件使用层次结构来描述三维对象。

32.1.1 VRML 的历史

从开始在互联网上发布文档，人们就试图通过采用高级三维图形并且与这些图形交互来提高网页内容的质量。VRML 这个术语是 Tim Berners-Lee 1994 年在欧洲环球网研讨会上谈论建立三维环球网标准的必要性时提出的。此后很快地，一个由艺术家和工程师组成的活跃团体形成了一个名为 www-vrml 的邮件发送清单。他们将标准的名称改为虚拟现实模型语言（Virtual Reality Modeling Language，VRML），以此强调图形学的地位。他们努力的结果是形成了 VRML 1.0 规范。作为该规范的基础，他们使用了 SGI 公司 Inventor 文件格式的一个子集。

VRML 1.0 标准可以在几种 VRML 浏览器上执行，但是使用它只能创建静态的虚拟世界。这个局限性使它不能推广使用。显然，这个语言需要添加动画和交互性方面的功能，以便给虚拟世界带来活力。后来 VRML 2.0 标准开发出来了，并且在 1997 年被采纳为国际标准 ISO/IEC 14772-1:1997，所以又称为 VRML97。

VRML97 是创建交互式三维场景（虚拟世界）的一个开放灵活的平台。随着计算机计算和绘图能力的飞速提高，以及通信更加快速，三维图形在艺术和游戏等传统领域以外的应用越来越常见了。现在，在几个平台上都可以获得支持 VRML97 的浏览器。同样，可供选择的 VRML 生成工具也越来越多。另外，许多传统的图像软件包如 CAD 等现在也提供 VRML97 格式的导入和导出功能。

虚拟现实工具箱通过使用 VRML97 技术，给 MATLAB 用户提供了一个开放的三维可视化解决方案。它对于 VRML97 在科学计算和交互式三维动画等领域的广泛应用十分有益。

32.1.2　VRML 坐标系统

VRML 使用右手笛卡儿坐标系统。将右手的大拇指、食指和中指伸直，两两正交，则大拇指表示 x 轴，食指表示 y 轴（向上），中指表示 z 轴。笛卡儿坐标如图 32-1 所示。

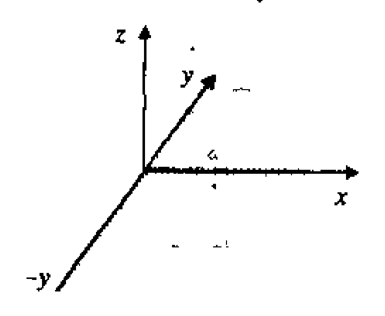


图 32-1　MATLAB 图形坐标系统

图 32-2　VRML 坐标系统

如图 32-2 所示，VRML 坐标系统与 MATLAB 的不同。MATLAB 坐标 z 轴向上为正，y 轴向后为正；而 VRML 坐标 z 轴向前为正，y 轴向上为正。

在 VRML 中，旋转角度用右手规则进行定义。如图 32-3 所示，用右手握住一根坐标轴，大拇指指向它的正向。剩下 4 个手指的指向为逆时针方向。这个方向就是对象旋转的正向。

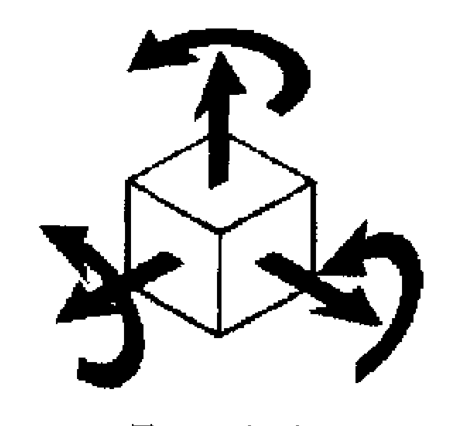


图 32-3　右手规则

在 VRML 文件的层次结构中，子对象的位置和方向相对于父对象进行指定。父对象有它自己的局部空间，该空间根据它的位置和方向定义。移动父对象同样会移动与它有关的子对象。

在 VRML 中，所有长度和距离以米为单位进行度量，角度单位为弧度。

32.1.3　VRML 数据类型

VRML 数据类型是 VRML 节点用于定义对象和 VRML 节点字段数据的数据类型。主要包括下面两个主题：

- VRML 字段数据类型；
- VRML 数据类类型。

1. VRML 字段数据类型

表 32-1 列出了所有 VRML 数据类型，以及它们与 MATLAB 类型的转换。

表 32-1　VRML 数据类型及其与 MATLAB 类型的对应关系

VRML 类型	描　述	虚拟现实工具箱中的类型
SFBool	布尔值，true 或 false	'on' 或'off'
SFFloat	32 位浮点值	Double
SFInt32	32 位有符号整型值 SFInt32_value = floor(double_value)	Double

续表

VRML 类型	描　述	虚拟现实工具箱中的类型
SFTime	绝对或相对时间值	Double
SFVec2f	有两个浮点值的向量，通常用于表示二维坐标，如纹理坐标	Double 型数组 (1×2)
SFVec3f	有三个浮点值的向量，通常用于表示三维坐标	Double 型数组(1×3)
SFColor	有三个浮点值的向量，用于指定 RGB 颜色	Double 型数组(1×3)
SFRotation	有四个浮点值的向量，用于指定坐标轴加上旋转角度的旋转坐标	Double 型数组(1×4)
SFImage	浮点值序列表示的二维数组	N/A
SFString	UTF-8 编码的字符串，与 ASCII 兼容，允许使用统一编码字符	String
SFNode	VRML 节点的容器	N/A
MFFloat	数组，元素为 SFFloat 值	Double 型数组(n×1)
MFInt32	数组，元素为 SFInt32 值	Double 型数组(n×1)
MFVec2f	数组，元素为 SFVec2f 值	Double 型数组(n×2)
MFVec3f	数组，元素为 SFVec3f 值	Double 型数组(n×3)
MFColor	数组，元素为 SFColor 值	Double 型数组(n×3)
MFRotation	数组，元素为 SFRotation 值	Double 型数组(n×4)
MFString	数组，元素为 SFString 值	字符单元数组
MFNode	数组，元素为 SFNode 值	N/A

2. VRML 数据类类型

一个节点可以包含 4 类数据：field、exposedField、eventIn 和 eventOut。这些类定义节点的行为，节点保存到计算机内存的方式和它们与其他节点和外部对象交互的方式。VRML 数据类如表 32-2 所示。

表 32-2　VRML 数据类

VRML 数据类	描　述
eventIn	节点可以接受的事件
eventOut	节点可以发送的事件
field	私有的节点成员，保存节点数据
exposedField	公有的节点成员，保存节点数据

（1）eventIn

通常，eventIn 事件对应于节点中的一个字段。节点字段不是从节点外部获取的，改变它们的惟一方法是存在对应的 enentIn。

有些节点的 eventIn 事件不对应于该节点的任何字段，但是为它提供了额外的函数。例如，Transform 节点有一个 addChindren eventIn，该事件被接收时，传递的子节点就添加到给定变换子对象的列表中。

（2）eventOut

不管什么时候，当允许发送事件的对应节点字段改变值时，本事件被发送。

（3）field

字段可被发送给 VRML 文件中的特定值。通常，字段对于节点来说是私有的，它的值只在节点接收一个对应的 eventIn 时发生改变。重要的是，要理解字段本身不会无缘无故地被其他节点或通过外部生成界面改变。

（4）exposedField

这是一个强大的 VRML 数据类，有很多用途。它用于同时具有 eventIn 和 eventOut 功能的字段类类型。对应 eventIn 的替代名称总是有 set-前缀，对应 eventOut 的字段名称总是有-changed 后缀。

exposedField 类定义对应 eventIn 和 eventOut 的行为。对于所有的 exposedField 类，当事件发生时，字段值发生改变，场景也发生相应的改变，eventOut 用一个新值进行传递。这样，在许多节点之间就可以发生连锁反应。

32.1.4 VRML 编辑工具

可以有多种方法创建 VRML 代码描述的虚拟世界。例如，可以直接用文本编辑器编写 VRML 代码，或者可以在不必懂得任何 VRML 语言方面基础知识的情况下用 VRML 编辑器创建虚拟世界。但是，有必要理解 VRML 树的结构，以连接虚拟世界和 Simulink 模块和信号。

32.1.5 VRML 文件格式

使用 VRML 生成工具创建虚拟世界不需要任何 VRML 格式方面的基础知识。但是，了解 VRML 场景描述方面的基础知识是有益的。这样，可以帮助你更有效地创建虚拟世界，而且对于虚拟世界中的元素是如何通过虚拟现实工具箱进行控制的有更好的理解。

VRML 中，一个三维场景用对象（或节点）的树状层次结构进行描述。树中的每一个节点表示场景的某些功能。有 54 种不同类型的节点，其中有些是形状节点（表示真实的三维对象），有些是组节点，用于聚合它的子节点。比如，下面有几种节点：

- Box 节点　表示场景中的一个箱形对象。
- Transform 节点　定义位置、比例、比例方向、旋转、平移和它的子树（组节点）的子对象。
- Material 节点　对应于场景中的材质。
- DirectionalLight 节点　表示场景中的光。
- Fog 节点　表示场景中的雾。
- ProximitySensor 节点　该节点使得可以与 VRML97 进行交互。当用户进入指定区域，或者在指定区域中存在或移动时产生事件。

每个节点包含了一列字段，它们的值定义节点的函数参数。

这些节点可以数在树的顶层或作为其他节点的子对象放在树层次中间。改变某节点字段中的值时，它的子树中所有节点都要受到影响。利用这个特点，可以在复杂的组合对象中定义相对位置。

在 VRML 场景定义语法中使用 DEF 关键字，可以用指定的名称标注每个节点。例如，语句

```
DEF MyNodeName Box
```

将一个 Box 节点的名称设素为 MyNodeName。

下面是一个简单的 VRML 文件，它建立的一个三维场景，其中有两个图形对象模型：

地板上有一个红色的球，地板用展平的 Box 节点对象表示。注意，VRML 是可读的文本文件，可以在任何文本编辑器中进行编辑。

```
#VRML V2.0 utf8
# This is a comment line
WorldInfo {
  title "Bouncing Ball"
}
Viewpoint {
  position 0 5 30
  description"Side View"
}
DEF Floor Box {
  size 6 0.2 6
}
DEF Ball Transform {
  translation 0 10 0
  children Shape {
    appearance Appearance {
      material Material {
        diffuseColor 1 0 0
      }
    }
    geometry Sphere {
    }
  }
}
```

第 1 行是 VRML 头行。每个 VRML 文件必须有这个头行。它说明这是一个 VRML 2.0 文件，并且文件中的文本对象按照 UTF8 标准进行编码。用符号#进行注释。除了第 1 行以外，每个以#符号打头的行都会被 VRML 查看器忽略。



图 32-4　VRML 定义的一个场景

本例中，大部分属性都设置为默认值，只重新指定了名称和维。为了能控制球的位置和其他属性，它被定义为一个 Transform 类型节点的子节点。这里，默认的单位圆球指定为红色，位于地板上方 10 的地方。另外，虚拟世界的标题被 VRML 查看器用于分辨不同的虚拟世界。VRML 文件中还定义了一个合适的初始视点。

这个 VRML 文件定义的场景如图 32-4 所示。

32.2　V-Realm 编辑器

32.2.1　VRML 编辑工具

VRML 文件使用标准的文本格式，可以用任何文本编辑器进行阅读。阅读文本对于调试、自动处理和直接改变代码很有帮助。同样，如果使用正确的 VRML 语法，可以用任何通用的文本编辑器创建虚拟场景，就象创建 HTML 网页一样。

很多人更喜欢用他们喜欢的文本编辑器创建简单的虚拟世界。但是，创建虚拟世界的主要方法还是使用三维编辑工具。利用这些工具，可以在对 VRML 语言没有深入了解的情况下创建比较复杂的虚拟场景。

这些三维编辑器为创建许多实用技术模型提供了必要的能力。例如，可以从一些 CAD 软件包导入三维对象，使得生成过程更简单更有效率。对于 VRML 生成，主要有两类三维编辑工具：

- 可以导出到 VRML 格式的常用三维生成软件包；
- 本地 VRML 生成工具。

常用的三维编辑器不将 VRML 作为默认格式，但可以另存为 VRML 格式。有很多商业软件包，如 3D Studio，具有这种功能。这些工具功能很多而且很容易使用。常见的三维编辑器面对特定专业人群开发，例如面对艺术、动画、游戏或科技应用等。根据应用领域的不同，它们提供了不同的工作环境。这些常用三维编辑器中，有些会很强大、很昂贵，并且不好学，而有些相对便宜并能满足特定的需求。

有趣的是，很多常用商业三维编辑器的图形用户界面与本地 VRML 生成工具的在很多方面很相像。例如，除了用不同图形学方法显示三维场景以外，它们都提供了层次树形视图和进行三维元素定义的快捷方式。

本地 VRML 编辑器将 VRML 作为编辑器自己的格式。这就保证了编辑器中的所有图形都符合 VRML 规范。不幸的是，目前达到商业化级别的高级 VRML 编辑器很少。大部分本地 VRML 编辑器还处在开发阶段，而且比常用三维编辑器更难使用。Ligo 公司开发的 V-Realm 编辑器是一个例外。它是目前在个人计算机上可以使用的最高级的 VRML 编辑工具之一。V-Realm 编辑器只在 Windows 操作系统上可用。

32.2.2　V-Realm 编辑器的安装

安装虚拟现实工具箱时，V-Realm 编辑器的文件已经拷贝到硬盘上了，但安装还没有完成。安装 VRML 编辑器会写一个密匙到 Windows 注册表中，这样就可以使用 V-Realm 编辑器中的其他库文件，并且使虚拟现实工具箱中的“Edit”按钮与本编辑器相连。

按照下面的步骤安装 VRML 编辑器。

（1）启动 MATLAB。

（2）在 MATLAB 命令窗口中输入

```
vrinstall -install editor
```

或输入

```
vrinstall('-install','editor')
```

MATLAB 显示下面的信息：

```
Starting editor installation...
Done.
```

（3）输入

```
vrinstall -check
```

如果编辑器安装成功，MATLAB 会显示下面的信息。

```
VRML editor:      installed
```

32.2.3 设置虚拟场景的默认编辑器

可以用一个 V-Realm 编辑器或任何文本编辑器编辑虚拟场景，文本编辑器中通过 VRML 语言进行编辑。使用 vrsetpref 和 vrgetpref 命令，可以将编辑器用于编辑场景。

下面的例子演示如何将编辑器从 V-Realm 编辑器变为文本编辑器。

（1）在 MATLAB 命令窗口中输入

```
vrinstall -check
```

确定是否安装了 V-Realm 编辑器，如果安装了，MATLAB 显示

```
VRML editor:      installed
```

（2）输入下面的命令行，确定默认编辑器。

```
a = vrgetpref
```

MATLAB 显示

```
a =
DefaultFigurePosition: [5 25 400 320]
         DefaultPanelMode: 'halfbar'
            DefaultViewer: 'web'
                   Editor: [1x60 char]
                 HttpPort: 8123
          TransportBuffer: 5
                   VrPort: 8124
```

变量 *a* 是一个结构数组，需要通过索引确定其 Editor 属性。

（3）确定默认编辑器，输入

```
a.Editor
```

MATLAB 显示

```
ans =
"%matlabroot\toolbox\vr\vrealm\program\vrbuild2.exe" "%file"
```

这是 V-Realm 编辑器可执行文件的路径。V-Realm 编辑器是当前 VRML 编辑器。

（4）确定 V-Realm 编辑器是默认的编辑器。在 MATLAB 命令窗口中输入

```
vrpend
```

（5）在 vrpend 模型窗口中，双击“VR Sink”块，打开“Block Parameters”对话框，如图 32-5 所示。

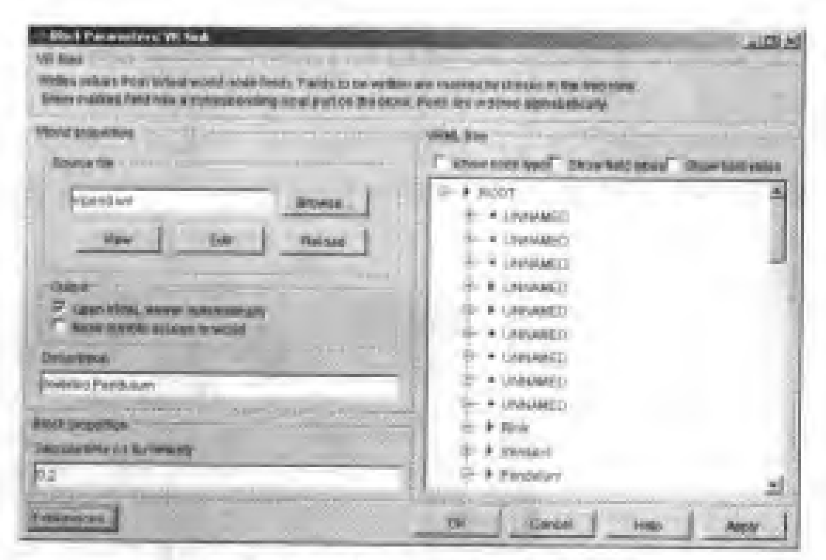


图 32-5 “Block Parameters”对话框

（6）单击“Edit”按钮。vrpend 模型在 V-Realm 编辑器中被打开，如图 32-6 所示。



图 32-6 V-Realm 编辑器

（7）输入下面的命令行，将默认的编辑器改变为 MATLAB 编辑器。

```
vrsetpref('Editor','%matlabroot\bin\win32\meditor.exe %file')
```

指定编辑器可执行文件的路径，可以将任何想用的编辑器设置为当前编辑器。

（8）重复 vrpend 演示，打开“VR Sink Block Parameters”对话框。

（9）单击“Edit”按钮，打开 MATLAB 编辑器，它现在被设置为默认的 VRML 编辑器。

（10）将 V-Realm 编辑器作为默认的 VRML 编辑器，输入

```
vrsetpref('Editor','factory')
```

现在单击“Edit”按钮将启动 V-Realm 编辑器。

32.2.4　V-Realm 编辑器的界面环境

安装虚拟现实工具箱时，V-Realm 编辑器的文件就被拷贝到硬盘中了。该编辑器的可执行文件位于 MATLAB 安装目录下的/toolbox/vr/vrealm/program 目录中，名称为 vrbuild2.exe。双击该文件的图标，可以打开编辑器的界面，如图 32-7 所示。



图 32-7　V-Realm 编辑器的界面

V-Realm 编辑器的图形界面不仅提供了三维场景的图形表示和进行图形元素交互创建的工具，还提供了虚拟世界中所有结构元素的层次树形视图。图 32-8 是打开一个虚拟世界以后的界面显示，其中左侧为树形视图，右侧为虚拟世界的图形显示。

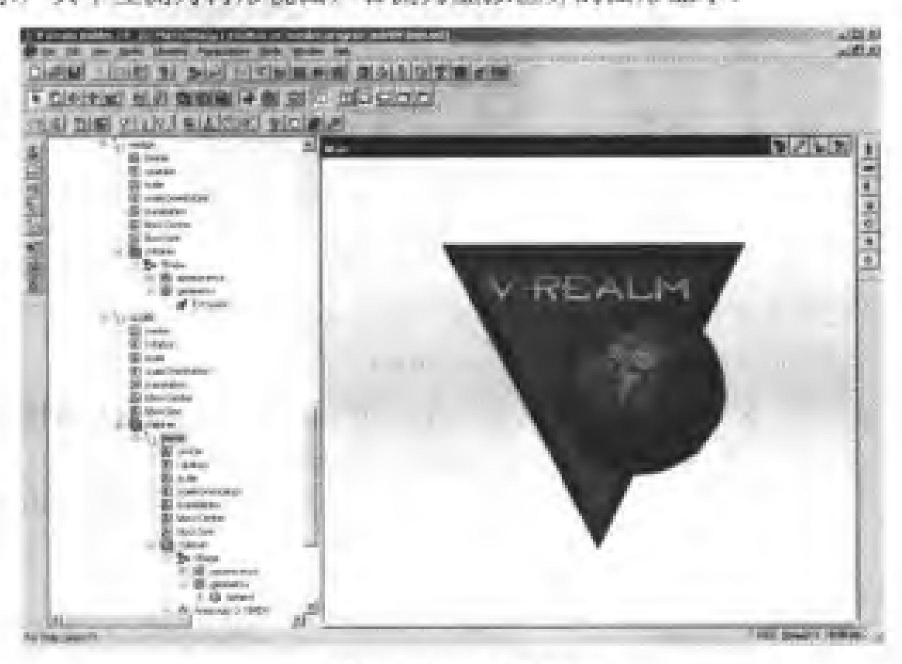


图 32-8　打开一个虚拟世界

虚拟世界中的这些结构元素称为节点。V-Realm 编辑器根据这些节点的类型列出它们和它们的属性，它支持所有 54 种 VRML97 类型。对于每种节点，有一个修改节点参数的便捷工具。可以有两种方法获取节点属性：

- 使用对话框（这些对话框可以从树形查看器中得到）；
- 直接使用定点设备。

多数情况下，使用树形视图更容易，因为在三维场景中选择一个指定的对象是比较困难的。使用树形视图，还可以很容易地改变某些节点的嵌套水平，以便根据自己的想法修改虚拟世界。在树形查看器中，可以给节点命一个惟一的名字——在虚拟现实工具箱中，它是必要的。

32.2.5 用 V-Realm 编辑器创建虚拟场景

用 V-Realm 编辑器可以创建虚拟场景，并在场景中添加几何对象、设置颜色、材质、光照和纹理等。

1. 节点

虚拟场景在数据结构上是一个有向无环图，其中包括不同种类的场景对象，每一类对象可以称为场景的一个节点。如图 32-9 所示，V-Realm 编辑器中可以插入、编辑的节点包括外观、环境、体元、组合、光照和传感器等。



图 32-9 “Nodes”菜单

2. 三维图形元素

三维图形元素（体元）是构建三维场景模型的基本要素。如图 32-10 所示，V-Realm 编辑器提供了立方体、锥体、圆柱、球体、文本、网格、自由台体等体元。

图 32-10 V-Realm 编辑器提供的体元

图 32-11 和图 32-12 演示了一个台体的创建过程。在工具条中单击创建台体的命令按钮以后，会在绘图区显示一个六面体。在树形视图中找到“Extrusion”项目，双击它，弹出“Extrusion Editor”对话框，如图 32-11 所示。在该对话框中可以编辑台体的横断面，利用工具按钮，可以设置三角形、圆形和正方形三种基本断面，拖拉横断面上的手柄，可以创建任意形状的断面。单击“确定”按钮，在绘图区显示对应横断面形状的三维台体，如图 32-12 所示。



图 32-11　"Extrusion Editor" 对话框



图 32-12　对应横断面的台体



图 32-13　"Select Object" 对话框

除了利用体元自己创建三维对象外，V-Realm 编辑器还提供了一个对象库，可以将库中的对象导入到场景中来。图 32-13 是导入对象的 "Select Object" 对话框。

3．颜色

在场景中右键单击对象，然后在弹出的菜单中单击 "Color" 选项，打开 "Color Mode Painter" 对话框，如图 32-14 所示。在对话框中可以设置漫反射光、环境光和镜面光的颜色，以及环境光的强度、颜色的亮度、透明度等。

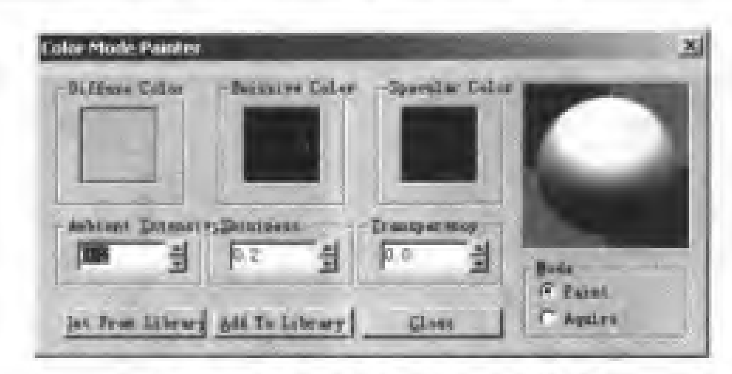


图 32-14 "Color Mode Painter" 对话框

4．光照、材质和纹理

光照对于场景效果非常重要，分别单击工具条上的按钮，可以设置镜面光、点光和聚光。V-Realm 编辑器提供了材质库和纹理库，可以直接从库中导入对象的材质和纹理。导入材质使用 "Select Material" 对话框，如图 32-15 所示；导入纹理使用 "Select Texture" 对话框，如图 32-16 所示。



图 32-15 "Select Material" 对话框



图 32-16 "Select Texture" 对话框

32.2.6 用 V-Realm 编辑器编辑虚拟场景

创建与编辑的过程是共同进行的，创建是目的，编辑是过程。V-Realm 编辑器提供了比较完善的交互编辑功能。

例如图 32-17 中，点击球体对象时，显示球体的包围盒，并在球心坐标轴方向上显示带有绿色圆圈的引线。拖拉包围盒上的白色手柄，可以缩小或放大球体，如图 32-18 所示；拖拉引线上的绿色圆圈，可以旋转球体，如图 32-19 所示。

旋转球体还有另一种常见形式，在球体的右击弹出式菜单中，在 "Manipulators" 子菜单中单击 "Center ball" 选项，可以设置滚动球操作。如图 32-20 所示，在图中拖拉鼠标时，球随



图 32-17 球体的包围盒和引线

着鼠标绕球心滚动。

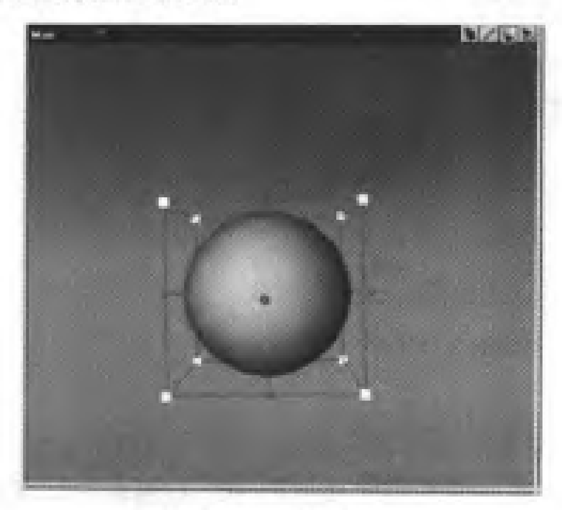


图 32-18 放大球体

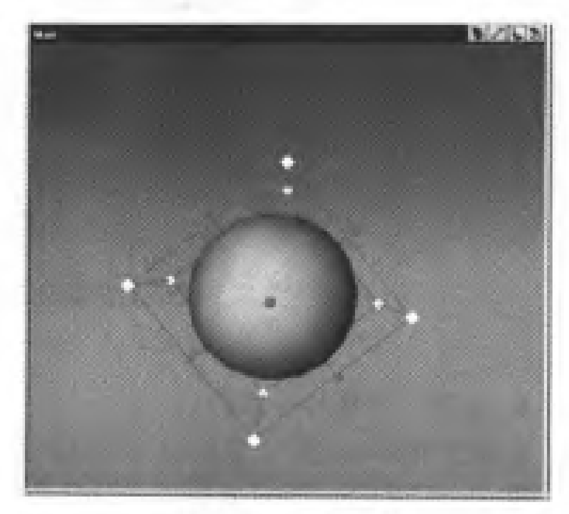


图 32-19 旋转球体

通过设置，还可以查看对象的线框模型。图 32-21 中是球体的线框模型。

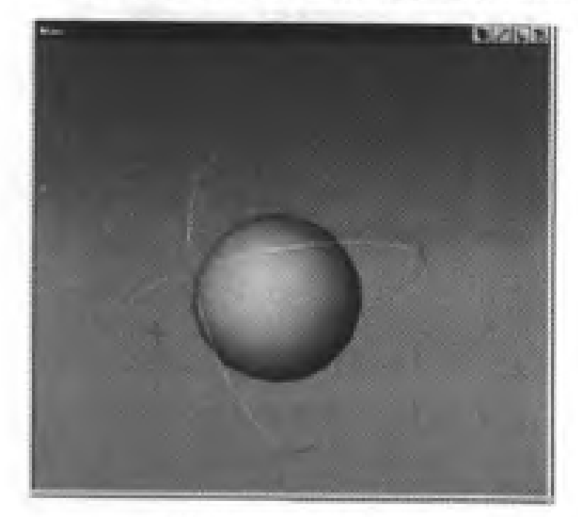


图 32-20 跟踪球技术

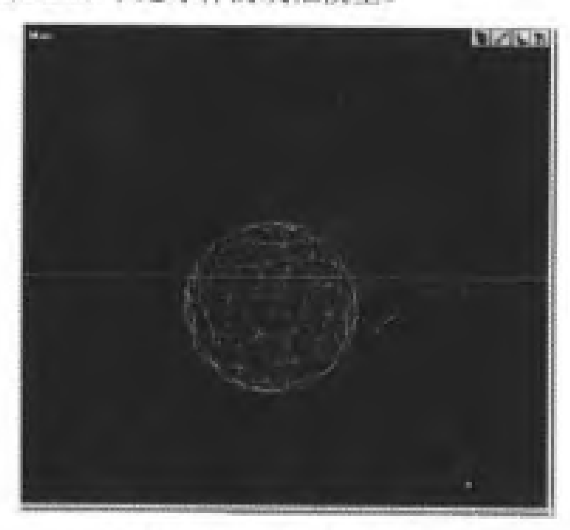


图 32-21 球体的线框模型

第 33 章　MATLAB 与虚拟世界进行交互

用 V-Realm 编辑器建立的虚拟世界模型可以显示在一定的工具界面中，然后通过在 MATLAB 命令窗口中直接输入命令或使用 M 文件来控制虚拟世界。

33.1　显示虚拟世界

显示虚拟世界的工具界面可以是虚拟现实工具箱提供的 VRML 查看器，也可以是微软的因特网浏览器或网景的浏览器。使用后者需要使用 blaxxun Contact 插件。

33.1.1　VRML 查看器

安装虚拟现实工具箱时，将虚拟现实工具箱提供的 VRML 查看器设置为默认的查看器。VRML 查看器如图 33-1 所示。虚拟世界显示在主窗口中，利用窗口下方的各种按钮可以进行平移、旋转等操作。

右键单击虚拟世界中的对象，显示编辑菜单，如图 33-2 所示。利用菜单中的选项，可以设置显示哪个虚拟世界、控制面板的显示方式、场景漫游的方法和渲染方式等。可以设置场景对象的反走样、光照、纹理和线框模型等。



图 33-1　VRML 查看器



图 33-2　用弹出式菜单编辑虚拟世界

33.1.2　网络浏览器

也可以使用网络浏览器查看 VRML 虚拟世界。使用浏览器查看，需要首先安装 VRML 插件，下面主要介绍 blaxxun Contact 插件。

1．安装 VRML 插件

安装虚拟现实工具箱时，将虚拟现实工具箱查看器设置为默认的查看器。如果想将 Web 浏览器作为 VRML 浏览器，按照下面的步骤安装 blaxxun Contact 插件。可以将这个插件与微软的 IE 浏览器或网景的浏览器一起使用。

注意，blaxxun Contact 安装程序只将插件安装在当前浏览器上。如果改变默认的浏览器，需要进行第 2 次安装。blaxxun Contact 的可执行文件位于 C:\<MATLAB root>\toolbox\vr\blaxxun 目录中，其中，<MATLAB root>为 MATLAB 的根目录。

跟虚拟现实工具箱的 3.1 版本一起使用，必须使用 blaxxun Contact 插件的 4.4 版本。这个版本的插件与虚拟现实工具箱一起发布。可以从网址 http://www.mathworks.com/support/product/VR/下载。

如果安装了 MATLAB 的 Web 服务器，要确保安装插件前 Web 服务器已经停止使用。同样，进行以下安装步骤以前，要确定已经连接到因特网。

（1）启动 MATLAB。

（2）在 MATLAB 命令窗口中输入

```
vrinstall –install viewer
```

MATLAB 显示下面的提示信息，询问使用 OpenGL 加速还是 Direct3d 加速：

```
Do you want to use OpenGL or Dirept3d acceleration? (o/d)
```

（3）输入 o 或 d，确定使用哪一种加速方法。如果不能确定，输入 d，选择 Direct3d 加速。

```
d
```

blaxxun 安装程序开始运行，并显示图 33-3 所示的“blaxxun Contact-Welcome”对话框。

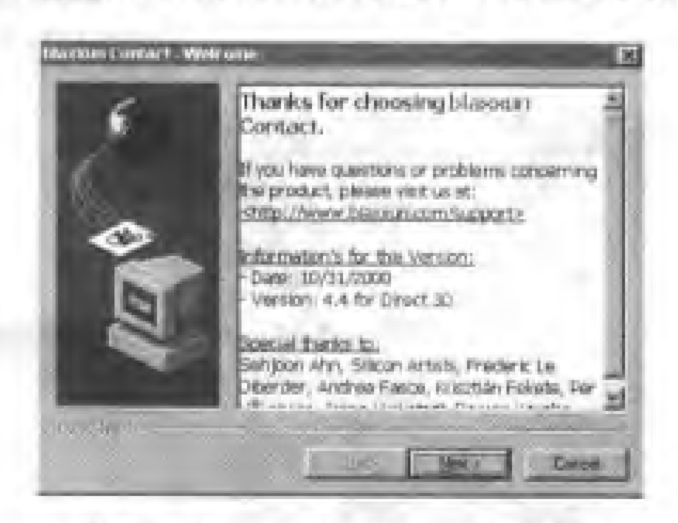


图 33-3　“blaxxun Contact-Welcome”对话框

（4）按照提示完成后面的安装设置。

（5）在MATLAB命令窗口中输入

```
vrinstall -check
```

如果安装成功，MATLAB会显示下面的信息。

```
VRML viewer:    installed
```

如果安装没有成功，显示没有安装的信息。

```
VRML viewer:    not installed
```

2. 使用blaxxun Contact插件时存在的问题

blaxxun Contact插件与虚拟现实工具箱3.1和微软的因特网浏览器5.5以上版本一起使用时，可能会出现不能更新虚拟场景的问题。网景用户没有遇到这个问题。

如果使用因特网浏览器5.5以上版本，与虚拟现实工具箱3.1版本一起使用blaxxun Contact 4.4以前必须更改网络安全设置。升级blaxxun Contact的版本不能解决这个问题。

按照下面的步骤改变网络的安全设置。

（1）打开因特网浏览器。

（2）在"Tool"菜单中选择"Internet Options"选项，打开"Internet Options"对话框。

（3）单击"Security"标签。

（4）选择"Custom Level"按钮，打开"Security Settings"对话框。

（5）找到"Microsoft VM"选项。第1个副标题是"Java permissions"。

（6）选择"Custom"，"Security Settings"对话框的左下角显示"Java Custom Settings"按钮。

（7）单击"Java Custom Settings"按钮，打开"Local intranet"对话框。

（8）单击"Edit Permissions"标签。

（9）在标题和副标题中找到"Run Unsigned Coutent"。

（10）在"Run Unsigned Content"下方找到"Access to all Network Addresses"。

（11）在"Access to all Network Addresses"下方选择"Enable"按钮。

（12）单击"OK"按钮，"Local intranet"对话框关闭。

（13）在"Security Settings"对话框上单击"OK"按钮。系统会问你是否改变安全设置。

（14）单击"Yes"按钮。

（15）在"Internet Options"对话框中选择"OK"按钮。

3. 用浏览器显示虚拟世界

安装blaxxun Contact插件以后，就可以用浏览器显示虚拟世界了。用IE浏览器打开<MATLABA>\toolbox\vr\vrdemos目录下的vrmount文件，显示虚拟世界如图33-4所示。

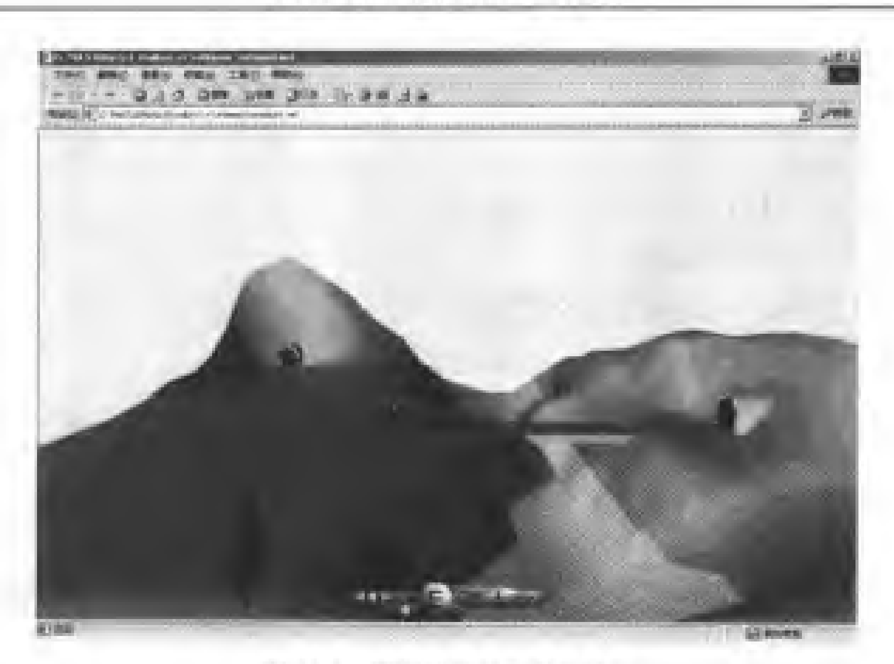


图 33-4　用浏览器显示虚拟世界

33.2　与虚拟世界交互

在命令窗口中使用命令行可以实现与虚拟世界的交互。

33.2.1　创建虚拟现实工具箱对象

通过 MATLAB 命令行界面与虚拟世界进行交互以前，需要创建 vrworld 对象。不能直接与虚拟世界交互。虚拟世界在 VRML 文件中定义，文件扩展名为.wrl。创建虚拟世界以后，就可以创建一个 vrworld 对象。下面使用虚拟世界 vrmount.wrl 进行演示。

（1）打开 MATLAB，在命令窗口输入

```
myworld = vrworld('vrmount.wrl')
```

MATLAB 显示

```
myworld =
     vrworld object: 1-by-1
```

（2）输入

```
vrwhos
```

MATLAB 显示以下信息

```
VR Car in the Mountains
     Loaded from 'D:\MATLAB6p5p1\toolbox\vr\vrdemos\vrmount.wrl'.
     Visible for local viewers.
     No clients are logged on.
     World id is 'W1082896508612'.
```

vrworld 对象 myworld 与虚拟世界 vrmount.wrl 相联系。可以认为变量 myworld 是 vrworld 对象的一个句柄。

33.2.2 使用MATLAB接口

本节包括以下内容：

- 打开虚拟世界——打开一个虚拟世界并扫描它的结构。
- 与虚拟世界进行交互——为可以获取的虚拟世界节点和它们的字段设置新值。
- 关闭和删除vrworld对象——关闭打开的虚拟世界并将它们从内存中删除。

1．打开虚拟世界

通过打开虚拟世界，可以在VRML查看器中查看虚拟世界，扫描它的结构并从MATLAB命令窗口中改变虚拟世界的属性。

创建vrworld对象以后，可以用与该虚拟世界相连的vrworld对象打开虚拟世界。下面打开与虚拟世界vrmount.wrl相连的vrworld对象myworld。

（1）在MATLAB命令窗口输入

```
open(myworld);
```

MATLAB打开虚拟世界vrmount.wrl。

（2）输入

```
set(myworld, 'Description', 'My first virtual world');
```

描述属性变为“My first virtual world.”。这个描述显示在所有的虚拟现实对象列表中，显示在虚拟现实工具箱查看器的标题条中和虚拟现实工具箱HTML页上虚拟世界的列表中。

（3）打开Web浏览器。在域名窗口中输入

```
http://localhost:8123
```

浏览器通过用一个与“My first virtual world”的链接显示虚拟现实工具箱HTML页。数字8123是默认的虚拟现实工具箱端口号。该端口号可以改变。

如果该Web浏览器支持VRML，在浏览器窗口中单击“My first Virtual world.”。浏览器中将显示虚拟世界vrmount.wrl。

另外，也可以用命令view(myworld)显示虚拟世界，它将虚拟场景显示在默认的查看器中。

2．与虚拟世界交互

可以用vrnode对象方法将所有可以获取的虚拟世界节点和它们的字段设置为新值。利用这个方法，可以在MATLAB环境中改变和控制虚拟世界的自由度。

vrworld类型的对象包含有多个节点，这些节点是vrnode类型的，在VRML文件中用DEF语句进行命名。

打开vrworld对象以后，就可以获取虚拟世界中可用的节点列表。下面利用vrworld对象myworld和虚拟世界vrmount.wrl进行演示。

（1）在MATLAB命令窗口输入

```
nodes(myworld);
```

MATLAB会显示vrnode对象及其字段的列表：

```
View1 (Viewpoint) [My first virtual world]
Camera_car (Transform) [My first virtual world]
```

```
VPfollow (Viewpoint) [My first virtual world]
Automobile (Transform) [My first virtual world]
Wheel (Shape) [My first virtual world]
Tree1 (Group) [My first virtual world]
Wood (Group) [My first virtual world]
Canal (Shape) [My first virtual world]
ElevApp (Appearance) [My first virtual world]
River (Shape) [My first virtual world]
Bridge (Shape) [My first virtual world]
Road (Shape) [My first virtual world]
Tunnel (Transform) [My first virtual world]
```

（2）输入

```
mynodes = get(myworld, 'Nodes')
```

MATLAB 创建一个 vrnode 对象数组，这些对象对应于虚拟世界节点和显示。

```
mynodes =
    vrnode object: 13-by-1

    View1 (Viewpoint) [My first virtual world]
    Camera_car (Transform) [My first virtual world]
    VPfollow (Viewpoint) [My first virtual world]
    Automobile (Transform) [My first virtual world]
    Wheel (Shape) [My first virtual world]
    Tree1 (Group) [My first virtual world]
    Wood (Group) [My first virtual world]
    Canal (Shape) [My first virtual world]
    ElevApp (Appearance) [My first virtual world]
    River (Shape) [My first virtual world]
    Bridge (Shape) [My first virtual world]
    Road (Shape) [My first virtual world]
    Tunnel (Transform) [My first virtual world]
```

（3）输入

```
whos
```

MATLAB 显示下面的信息：

```
Name          Size                    Bytes    Class
  mynodes     13x1                     3824 vrnode object
  myworld      1x1                      152 vrworld object
```

现在可以获取节点特征并给某些节点属性设置新值了。例如，可以用“Automobile”改变汽车的位置，“Automobile”是虚拟世界中的第 4 个节点。

（4）输入下面的语句，取得 Automobile 节点的字段。

```
fields(mynodes(4));
```

或者输入

```
fields(myworld.Automobile)
```

MATLAB 显示下面的列表：

```
Field                    Access    TypeSync
------------------------------------------------
translation          exposedFieldSFVec3foff
center               exposedFieldSFVec3foff
bboxCenter           field     SFVec3foff
children             exposedFieldMFNodeoff
scale                exposedFieldSFVec3foff
bboxSize             field     SFVec3foff
removeChildren       eventIn   MFNodeoff
scaleOrientation     exposedFieldSFRotationoff
rotation             exposedFieldSFRotationoff
addChildren          eventIn   MFNodeoff
```

Automobile 节点是 Transform 类型的。通过改变 translation 字段的值，可以改变该节点的位置。从上面的列表中，可以看出，translation 字段需要 3 个值，分别表示对象的[x y z]坐标值。

（5）输入

```
view(myworld)
```

默认的查看器打开和显示虚拟世界 vrmount.wrl，如图 33-5 所示。



图 33-5 打开和显示虚拟世界 vrmount.wrl

（6）并排移动 MATLAB 窗口和浏览器窗口，这样可以同时查看它们。在 MATLAB 窗口中输入

```
myworld.Automobile.translation = [15 0.25 20];
```

MATLAB 给 Automobile 节点设置一个新的位置，可以发现，在 VRML 浏览器窗口中，汽车到了一个新的位置。

3. 关闭和删除 vrworld 对象

完成一个阶段的操作以后，必须关闭所有打开的虚拟世界并将它们从内存中删除。用下面的命令**关闭**和删除虚拟世界 myworld。

```
close(myworld);
delete(myworld);
```

这样，表示 vrworld 对象 myworld 的虚拟世界就从内存中删除了。所有与查看器和浏览器可能的连接都会关闭，并且虚拟世界的名称也从可获取虚拟世界名称列表中删除了。但是注意，关闭和删除一个虚拟世界并不将该虚拟世界对象的句柄从 MATLAB 工作空间中删除。

第 34 章　虚拟现实工具箱中的对象

第 33 章我们讲到了，MATLAB 与虚拟世界交互需要首先创建对象，这些对象与虚拟世界相对应，对对象进行操作实际上就是对虚拟世界进行操作。

34.1　vrworld 对象

vrworld 对象是虚拟场景的句柄。通过它，可以与场景交互并控制场景。

34.1.1　vrworld 对象的属性

利用 vrworld 对象的属性，可以控制对象的行为。表 34-1 列出了 vrworld 对象的属性。

表 34-1　vrworld 对象的属性

属　性	值	描　　述
Clients	Scalar	当前查看虚拟世界的客户数目。只读
ClientUpdates	off | on 默认值：on	客户不能或能更新虚拟场景。读/写
Description	String 默认值：自动取自 VRML 文件属性 Title	虚拟世界的描述。读/写
Figures	vrfigure 对象组成的向量	向量，元素为当前查看虚拟世界的虚拟现实工具箱查看器的句柄。只读
FileName	String	相关 VRML 文件的名称。读/写
Nodes	vrnode 对象组成的向量	虚拟世界中指定的 vrnode 的对象。只读
Open	off | on 默认值：off	表示一个关闭或打开的虚拟世界。只读
RemoteView	off | on 默认值：off	远程获取标记。读/写
View	off | on 默认值：on	表示一个不可查看或可以查看的虚拟世界。读/写

34.1.2　vrworld 对象的方法

利用 vrworld 对象的方法，可以获取和操作对象。表 34-2 列出了 vrworld 对象的所有方法。

表 34-2　vrworld 对象的方法

方　法	描　　述
vrworld	创建一个与虚拟世界相关的 vrworld 新对象
vrworld/close	关闭虚拟世界
vrworld/delete	从内存中删除虚拟世界
vrworld/edit	在外部 VRML 编辑器中打开一个虚拟世界
vrworld/get	读取 vrworld 对象的属性值
vrworld/isvalid	如果 vrworld 对象合法，返回 1；否则返回 0
vrworld/nodes	列出虚拟世界中可用的所有节点
vrworld/open	打开虚拟世界
vrworld/reloa	从相关 VRML 文件中重新载入虚拟世界
vrworld/save	将虚拟世界写到 VRML 文件
vrworld/set	改变 vrworld 对象的属性值
vrworld/view	查看虚拟世界

34.2　vrnode 对象

vrnode 对象是 VRML 节点的句柄。通过它，可以获取和设置节点属性值。它是 vrworld 对象的子对象。

34.2.1　vrnode 对象的属性

利用 vrnode 对象属性可以控制对象的行为。表 34-3 列出了 vrnode 对象的所有属性。

表 34-3　vrnode 对象的属性

属 性	值	描　　述
Fields	单元数组	VRML 节点的合法字段名。只读
Name	String	节点名称。只读
Type	String	节点的 VRML 类型。只读
World	句柄	父 vrworld 对象的句柄。只读

34.2.2　vrnode 对象的方法

利用 vrnode 对象的方法，可以获取和操作对象。表 34-4 列出了该对象的所有方法。

表 34-4　vrnode 对象的方法

方　　法	描　　述
vrnode	给已经存在的节点创建一个新的节点或句柄
vrnode/delete	删除一个 vrnode 对象
vrnode/fields	返回节点对象 VRML 字段的综述
vrnode/get	读取 vrnode 对象或 VRML 字段的属性值
vrnode/getfield	获取一个 vrnode 对象的字段值
vrnode/isvalid	如果 vrnode 对象合法，则返回 1；否则返回 0

续表

方　法	描　述
vrnode/set	改变虚拟世界节点的属性或 VRML 字段
vrnode/setfield	改变 vrnode 对象的字段值
vrnode/sync	使客户的 VRML 字段同步或不同步

34.3　vrfigure 对象

vrfigure 对象是虚拟现实工具箱查看器窗口的句柄，通过它可以获取和设置查看器的属性。vrfigure 对象是 vrworld 对象的子对象。

34.3.1　vrfigure 对象的属性

利用 vrfigure 对象属性可以控制对象的行为。表 34-5 列出了 vrfigure 对象的所有属性。

表 34-5　vrfigure 对象的属性

属　性	值	描　述
Antialiasing	'off' | 'on' 默认值：off	确定渲染场景时是否使用了反走样。反走样通过在纹理点之间插值来平滑纹理。读/写
CameraBound	'off' | 'on' 默认值：on	控制相机是否以当前视点移动。读/写
CameraDirection	向量，元素为 3 个 double 型值	指定与当前视点相关的相机方向。读/写
CameraDirectionAbs	向量，元素为 3 个 double 型值	在世界坐标中指定相机方向。只读
CameraPosition	向量，元素为 3 个 double 型值	指定相对于当前视点的相机位置。读/写
CameraPositionAbs	向量，元素为 3 个 double 型值	指点相机在世界坐标中的位置。只读
CameraUpVector	向量，元素为 3 个 double 型值	指定相机相对于当前视点抬升向量的抬升向量。读/写
CameraUpVectorAbs	向量，元素为 3 个 double 型值	指定相机在世界坐标系中的抬升向量，只读
DeleteFcn	String	关闭 vrfigure 对象时进行回调。读/写
Headlight	'off' | 'on' 默认值：on	关/闭高光。读/写
Lighting	'off' | 'on' 默认值：on	打开/关闭场景的光照。没有光照，场景没有三维效果。读/写
Name	String	该 vrfigure 对象的名称。读/写
PanelMode	'opaque' 'translucent' 'off' 'halfbar' 'bar' 默认值：'halfbar'	控制虚拟现实工具箱查看器窗口中控制面板的外观。读/写
Position	向量，元素为四个 double 型值	该 vrfigure 对象的屏幕坐标。读/写
Textures	'off' | 'on' 默认值：on	执行/取消纹理渲染。读/写
Transparency	'off' | 'on' 默认值：on	指定进行渲染时是否考虑透明因素，读/写

续表

属　性	值	描　述
Viewpoint	String 如果活动视点没有名称，则值为空。	vrfigure 对象的活动视点，读/写
Wireframe	'off' | 'on' 默认值：off	指定对象绘制为实体还是线框。读/写
World	vrworld 对象	这个 vrfigure 对象对应的虚拟世界。只读
ZoomFactor	Double	相机缩放因子。读/写

34.3.2　vrfigure 对象的方法

vrfigure 对象是虚拟现实工具箱查看器窗口的句柄。利用它，可以获取和设置查看器的属性。该对象是 vrworld 对象的子对象。

表 34-6 列出了 vrfigure 对象的所有方法。

表 34-6　vrfigure 对象的方法

方　法	描　述
vrfigure	创建一个新的虚拟现实图像
vrfigure/capture	根据虚拟现实图像创建一幅 RGB 图像
vrfigure/close	关闭虚拟现实图像
vrfigure/get	读取 vrfigure 对象的属性值
vrfigure/isvalid	如果 vrfigure 对象是合法的，返回 1；否则返回 0
vrfigure/set	改变 vrfigure 对象的属性

第35章 虚拟现实工具箱中的函数

本章介绍虚拟现实工具箱中提供的交互函数。有些虚拟现实工具箱特性，如设置VRML文本值或数组等只能通过交互完成。

虚拟现实工具箱提供的函数不多，如表35-1所示。

表35-1 虚拟现实工具箱提供的函数

函　数	功　能
vrclear	从内存中删除所有关闭了的虚拟世界
vrclose	关闭虚拟现实图形窗口
vrdrawnow	更新虚拟世界
vrgetpref	读取虚拟现实工具箱的参数选项值
vrinstall	安装和检查虚拟现实工具箱组件
vrlib	为虚拟现实工具箱打开Simulink模块库
vrsetpref	改变虚拟现实工具箱的参数选项
vrview	用虚拟现实工具箱查看器或Web浏览器查看虚拟世界
vrwho	提供内存中存在的虚拟世界的列表
vrwhos	提供内存中存在的虚拟世界的详细列表

35.1 vrclear函数

vrclear函数实现从内存中删除所有关闭了的虚拟世界。该函数的语法格式为

```
vrclear
vrclear('force')
```

vrclear函数从内存中删除所有关闭了的虚拟世界，并使所有与之相关的vrworld对象无效。该命令不影响打开的虚拟世界。打开的虚拟世界包括那些从Simulink中载入的虚拟世界。使用该命令有以下好处：

- 确保下一次发生很耗费内存的操作以前空余内存数量最大。
- 进行一般的内存清理工作。

vrclear('force')命令从内存中清除所有虚拟世界，包括从Simulink中打开的世界。

35.2 vrclose函数

vrclose函数关闭虚拟现实图形窗口。语法格式为

```
vrclose
vrclose all
```

例如，输入下面的命令行，可以打开一系列虚拟现实图形窗口。

```
vrpend
vrbounce
vrlights
vrcrane
```

重置查看器窗口，使它们都可见。输入

```
vrclose
```

所有虚拟现实图形窗口从屏幕上消失。

35.3 vrdrawnow 函数

vrdrawnow 函数更新虚拟世界。其语法格式为

```
vrdrawnow
```

vrdrawnow 函数更新虚拟世界并对查看器中的场景作相应的改变。正常情况下，对场景所作的改变是按先后顺序进行的，并且视图在下面两种情况下发生改变：

- MATLAB 空闲了一段时间以后（没有 Simulink 模型在运行，没有 M 文件在执行）；
- 一个 Simulink 步骤已经结束。

35.4 vrgetpref 函数

vrgetpref 函数读取虚拟现实工具箱的参数选项值。其语法格式为

```
x = vrgetpref
x = vrgetpref('preference_name')
x = vrgetpref('preference_name','factory')
x = vrgetpref('factory')
```

其中，preference_name 表示要读取的参数选项值。下面解释各语法格式的意义：

x=vrgetpref　将虚拟现实工具箱中所有参数选项值返回到一个结构数组中。

x=vrgetpref('preference_name')　返回指定参数选项值。如果 preference_name 是一个参数选项名称的单元数组，则返回对应参数选项值的单元数组。

x=vrgetpref('preference_name', 'factory')　返回指定参数选项的默认值。

x=vrgetpref('factory')　返回所有参数选项的默认值。

定义了表 35-2 中所示的参数选项。

表 35-2　vrgetpref 函数的参数选项

参数选项	说　明
DefaultFigurePosition	设置虚拟现实工具箱查看器窗口的位置和大小
DefaultPanelMode	确定查看器中控制面板的外观。合法的值为'opaque','translucent', 'off','halfbar', 'bar'和'factory'。默认值为'halfbar'
DefaultViewer	指定哪个查看器用于查看虚拟场景。当参数选项设置为'internal'时，使用虚拟现实工具箱查看器；参数选项设置为'web'时，使用 Web 浏览器。默认时设置为'internal'

续表

参数选项	说　明
Editor	VRML 编辑器的路径。如果路径为空，则使用 MATLAB 编辑器
httpPort	用于在 Web 上通过 HTTP 获取虚拟现实服务器的 IP 端口号。如果改变此参数选项，则必须重新启动 MATLAB 才有效
TransportBuffer	在虚拟现实服务器与客户之间进行通信时传送缓冲器的长度
VrPort	用于在虚拟现实服务器和它的客户之间进行通信的 IP 端口。如果改变这个参数选项，则必须重启动 MATLAB 才有效

注意，httpPort、VrPort 和 TransportBuffer 参数选项影响虚拟世界基于 Web 的视图效果。DefaultFigurePosition 和 DefaultPanelMode 参数影响虚拟现实工具箱查看器。

DefaultPanelMode 参数选项控制虚拟现实工具箱查看器中控制面板的外观。例如，将该值设置为“translucent”会使控制面板看起来是透明的。

DefaultViewer 参数确定虚拟场景是显示在虚拟现实工具箱查看器中还是显示在 Web 浏览器中。如果设置为'internal'，则虚拟现实工具箱查看器是默认的查看器。如果设置为'web'，则带有 VRML 插件的 Web 浏览器是默认的查看器。

Editor 参数选项包含了 VRML 编辑器可执行文件的路径。使用 edit 命令时，虚拟现实工具箱用所有编辑 VRML 文件所需要的参数来运行 VRML 编辑器可执行文件。运行编辑器时，虚拟现实工具箱使用编辑器参数选项值，就像在命令行中键入它一样。有两个参数需要说明，如表 35-3 所示。

表 35-3　两个参数及其说明

%matlabroot	指 MATLAB 根目录
%file	指 VRML 文件名

例如，Editor 参数选项可能有下面的取值形式：

'%matlabroot\bin\win32\meditor.exe %file'

注意可执行文件名和 VRML 文件名周围的引用记号。如果该参数为空，则使用 MATLAB 编辑器。

HttpPort 参数选项指定用于 Web 连接的网络端口。该端口在 Web URL 中给出，如下所示：

http://server.name:port_number

该参数的默认值为 8123。

TransportBuffer 参数选项定义客户-服务器通信时信息窗口的大小。它确定客户和服务器之间一次最多能传递多少信息。

通常，参数选项的取值越大，动画运行越平稳，但是耗时越长。

该参数的数认值为 5，在大多数情况下，它是最优的。应该只在动画严重失真或反应时间严重滞后的情况下才改变它的值。

VrPort 参数指定虚拟现实工具箱服务器和它的客户之间进行通信的网络端口。正常情况下，该通信对于用户来说完全是不可见的。该参数的默认值为 8124。

35.5　vrinstall 函数

vrinstall 函数安装和检查虚拟现实工具箱的组件。其语法格式为

```
vrinstall('action')
vrinstall action
vrinstall('action','component')
vrinstall action component
x = vrinstall('action', 'component')
```

该函数用到的变量如表 35-4 所示。

表 35-4　vrinstall 函数变量的说明

action	该函数的操作类型。值包括-interactive,-selftest,-check,-install 和-uninstall
component	进行操作的组件名称。值包括 viewer 和 editor

使用本函数管理与虚拟现实工具箱有关的可选软件组件的安装。当前有两个这样的组件，即 VRML 插件和 VRML 编辑器。action 变量的取值说明如表 35-5 所示。

表 35-5　变量值及其描述

变 量 值	描　述
-selftest	检查虚拟现实工具箱的完整性。如果该函数报告错误，则应该重新安装虚拟现实工具箱。函数 vrinstall 自动完成自检工作
-interactive	检查已经安装的组件，然后显示可供选择的未安装组件列表
-check	检查可选组件的安装。如果给定组件已经安装，返回 1；否则返回 0。如果不指定组件，则显示一个关于组件及其状态的列表
-install	安装可选组件。该操作需要指定组件名。所有组件都可以用这个命令安装，但是，它们中的有一些（目前只有插件）需要用系统标准卸载程序进行卸载
-uninstall	卸载可选组件。该选项目前只对编辑器可用。注意，该操作不会将编辑器文件从安装目录中删除，它删除编辑器的注册信息。如果想卸载 VRML 插件，需要退出 MATLAB，并在控制面板中选择“添加/删除程序”选项进行删除

35.6　vrlib 函数

vrlib 函数为虚拟现实工具箱打开 Simulink 模块库。语法格式为

```
vrlib
```

虚拟现实工具箱的 Simulink 库有 6 个模块：VR Sink,VR Source,VR Placeholder,VR Signal Expander, Joystick Input 和 Magellan SpaceMouse。同样，可以从 Simulink 模块图中获取这些模块。在 Simulink 窗口中，从“View”菜单中单击“Show Library Browser”选项。

35.7　vrsetpref 函数

vrsetpref 函数改变虚拟现实工具箱的参数选项。语法格式为

vrsetpref('preference_name', preference_value)

vrsetpref('factory')

该函数中的变量说明如表 35-6 所示。

表 35-6　变量说明

preference_name	参数选项的名称
preference_value	参数的新值

本命令将给定的虚拟现实工具箱参数选项设置为给定值。将'factory'作为惟一变量时，所有参数重置为默认值。

35.8　vrview 函数

vrview 函数用虚拟现实工具箱查看器或 Web 浏览器查看虚拟世界。其语法格式为

vrview

x = vrview('filename')

x = vrview('filename','-internal')

x = vrview('filename','-web')

vrview 函数打开默认的 Web 浏览器并载入包含可用虚拟世界列表的虚拟现实工具箱 Web 页。

x=vrview('filename') 创建与.wrl 文件相关的虚拟世界，打开虚拟世界并将它显示到虚拟现实工具箱查看器或 Web 浏览器中。返回虚拟世界的句柄。

x=vrview('filename', '-internal') 创建与.wrl 文件相关的虚拟世界，打开虚拟世界并将它显示到虚拟现实工具箱查看器中。

x=vrview('filename', '-web') 创建与.wrl 文件相关的虚拟世界，打开虚拟世界并将它显示到 Web 浏览器中。

35.9　vrwho 函数

vrwho 函数提供内存中虚拟世界的列表。语法格式为

vrwho

x = vrwho

如果不指定输出参数，则 vrwho 函数将内存中虚拟世界的列表显示到 MATLAB 命令窗口中。如果指定一个输出参数，则 vrwho 函数将一个句柄向量返回给已经存在的 vrworld 对象。

35.10　vrwhos 函数

vrwhos 函数提供内存中虚拟世界的详细列表。语法格式为

vrwhos

该函数将内存中虚拟世界的列表显示到 MATLAB 命令窗口中，列表中有比较详细的描述。vrwho 和 vrwhos 的关系与 who 和 whos 之间的近似。

第 36 章　地图制作工具箱简介

使用地图制作工具箱，可以在 MATLAB 环境中读取、分析和显示地理信息。因为地球和大部分天体通常都是球形的，所以地理数据常常在球坐标系或椭球坐标系中定义。地球曲面上定义的距离、方位、面积甚至直线都与 MATLAB 笛卡儿坐标系中的不同。

将球体上的地理信息显示到平面上还需要特殊的绘图技巧。地图制作工具箱可以用简单的命令创建地图。使用工具箱提供的地图数据，可以创建详细的底图。在底图上可以绘制自己的结果。还可以导入高精度的地图数据，这些数据可以从政府或研究性网站上得到。

下面结合一系列实例概略介绍如何用地图制作工具箱绘图。

36.1　创建底图

地理数据通常显示在底图上，底图中包含有类似海岸线或地形线的基本特征信息。创建底图的第 1 步是选择合适的投影方法。投影指的是如何将球体表面的信息表示在平面上。将三维对象显示在二维表面上的艺术称为绘图学。到现在为止，绘图者已经提出了许多不同的投影方法。每一种都在比例、形状和方向的保真性方面进行折中处理。从某种意义上讲，绘图的艺术就是选择投影方法的艺术，选择的投影方法应该适合于当前任务。准备工作的第 2 步是选择和显示合适而且详细的地图数据。数据的详细程度和数量要与底图的覆盖范围相吻合，这一点很重要。

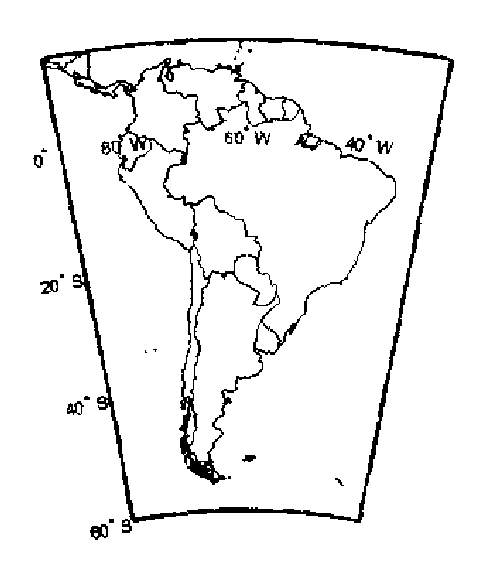


图 36-1　在地球底图中定义区域

尽管地图制作工具箱提供了很多投影方法和几种地图集数据，但利用它还是可以创建对大部分应用都合适的底图。创建地球底图的命令是 worldmap。该命令选择一种投影方法和适合指定区域的地图集部分数据。可以用某个洲或国家的名称，或纬度和经度范围定义区域，如果对名称的拼写没有把握，使用没有输入变量的 worldmap 命令，从显示的列表中选择名称。下面打开南美洲的底图：

```
worldmap 'south america'
hidem(gca)
```

生成图 36-1。

hidem(gca)命令指定打印时隐藏 MATLAB 的坐标轴。最好在打印以前就将坐标轴隐藏。

在本例中，worldmap 命令选择了低分辨率的地图集数据，因为南美洲覆盖的区域很大。该地图集数据包括海岸线、国界、湖和城市等。对于大面积区域的地图，例如南美洲地图，worldmap 命令选择低分辨率数据并忽略类似国家名

称和城市这样的细节。可以用可选参数覆盖 worldmap 命令的选项。

指定更小的区域时，worldmap 会转而使用高分辨率的地图集数据。图 36-2 是巴基斯坦的底图，它用面片代替了直线段。图中还添加了一个通用的绘图元素，即图形比例尺。

```
figure
worldmap('pakistan','patch')
scaleruler
hidem(gca)
```

生成图 36-2。可以用鼠标将比例尺拖拉到更好的位置。

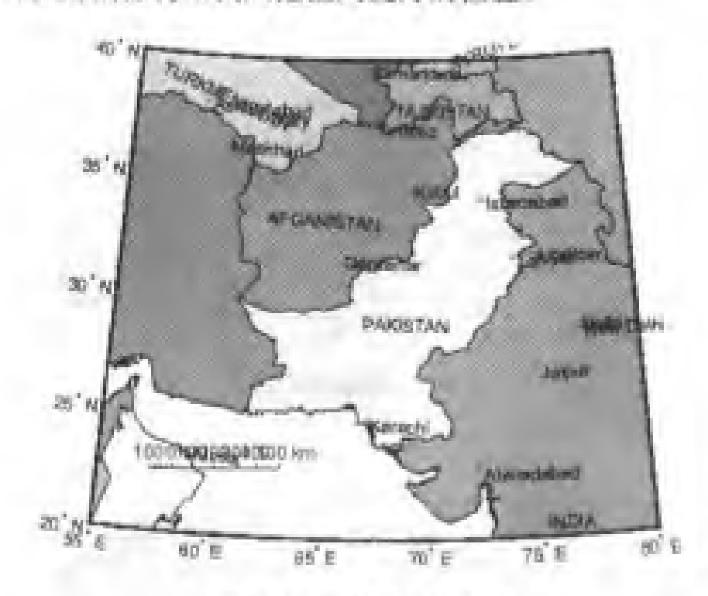


图 36-2　巴基斯坦的底图

还可以用纬度和经度范围指定目标区域。下面的例子创建波斯湾的底图。它还显示了如何使用 MATLAB 的句柄图形和地图制作工具箱的命令改变图形的属性，如国家的颜色等。

```
figure
worldmap([20 35],[45 65],'patch')
scaleruler
set(handlem('allpatch'),'Facecolor','w')
setm(gca,'FFaceColor','c')
hidem(gca)
```

生成图 36-3。

还可以用矩阵数据创建底图。地图制作工具箱定义了两种矩阵数据。正则矩阵地图显示在等边网格上，行和列与东西方向和南北方向对齐。一般的矩阵地图可以有任意间隔和对齐方式的元素。

下面是整个世界的正则矩阵地图示例。geoid 工作空间文件包含一个一度网格上的高程矩阵数据。该文件可以看作是没有海浪、潮汐和陆地影响的海面。将 worldmap 作为一个正则矩阵地图提供时，地理范围与矩阵的范围有关。这里，geoid 高程数据显示为彩色表面。contourcmap 命令创建一个颜色查找表，该表的色阶与等值线的水平对齐，使数据更容易从图像上直接读出来。

```
load geoid
figure
worldmap(geoid,geoidlegend)
contourcmap(10,'jet','colorbar','on','location','horizontal')
```

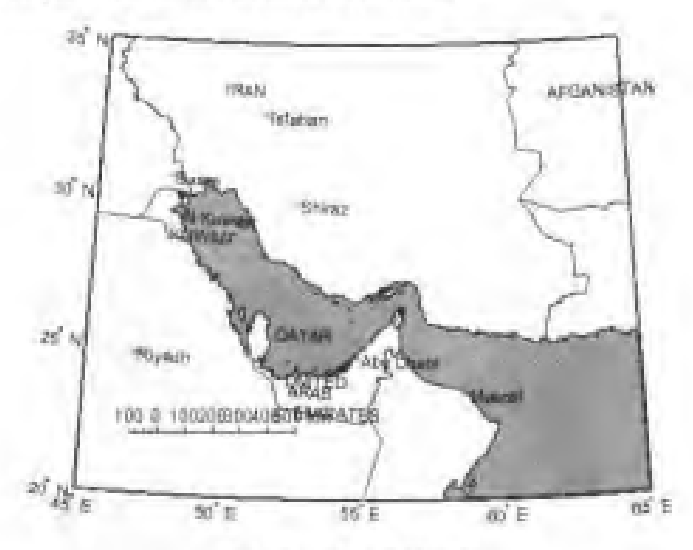


图 36-3　波斯湾的底图

生成图 36-4。

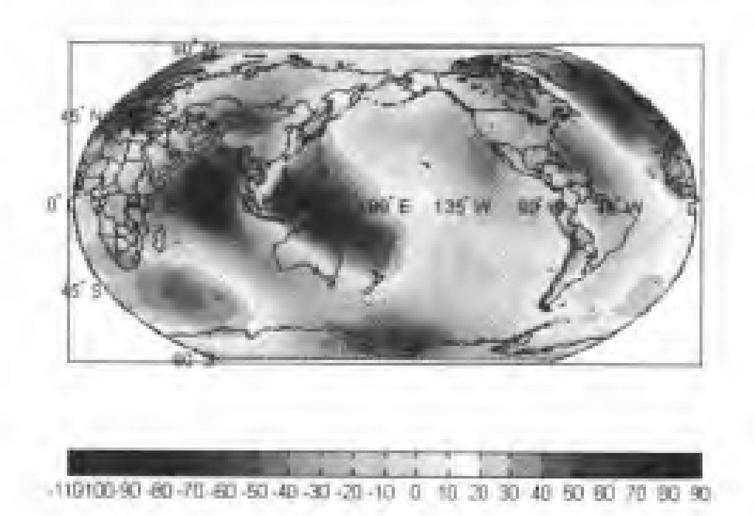


图 36-4　世界正则矩阵地图

其他颜色常用于显示高程矩阵。这种类型的数据通常称为数字高程模型（DEM）。下面的例子用三维光照表面显示了朝鲜半岛的高程和海深。在垂向上作了很大的拉伸，以使突出地形特征。用 daspectm 命令控制垂向比例。

```
load korea
figure
worldmap(map,maplegend,'ldem3d')
view(3)
```

生成图 36-5。

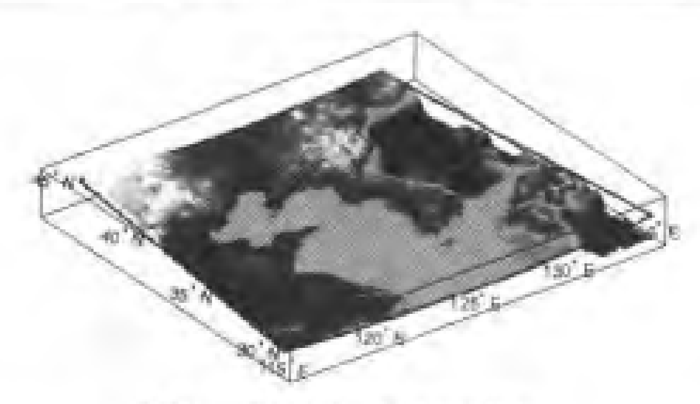


图 36-5　朝鲜半岛的数字高程模型

使用类似的 usamap 命令创建美国的底图。如果只显示相连的州，使用 usamap conus 命令。

```
figure
usamap conus
```

生成图 36-6。



图 36-6　美国的底图

要用更高分辨率的地图集数据显示更小的区域，使用州名或纬度和经度范围。

```
figure
usamap([37.5 40],[-78 -75])
hidem(gca)
```

生成图 36-7。

需要对投影和底图数据有更多控制时，用低级 axesm 命令代替 worldmap 或 usamap 命令。axesm 命令用于准备地图坐标系和设置初始投影参数。然后可以添加选择的地图集数据。下面使用正交投影，用低分辨率填充的国家轮廓显示底图。

```
figure
axesm('MapProjection','ortho','Origin',[-35 70 45])
framem;gridm
displaym(worldlo('POpatch'))
polcmap
hidem(gca)
```

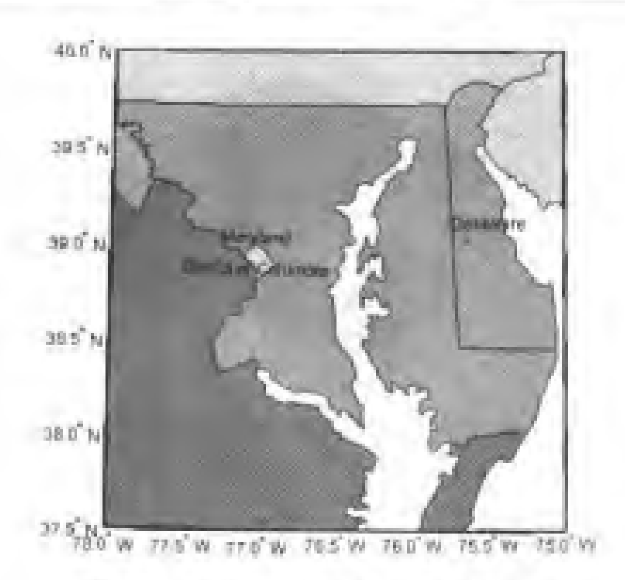


图 36-7　用高分辨率数据显示更小的区域

生成图 36-8。



图 36-8　使用正交投影

36.2　在底图上显示数据

地图制作工具箱可以显示向量、矩阵和结构类型的地理数据。工具箱中的命令名通常与 MATLAB 中的图形命令相似，只是在末尾添加了一个“m”。这个“m”表示数据是地理数据，并且要用已经定义的地图投影进行显示。例如，在 MATLAB 中，使用 plot(x,y)，将一个点向量显示为直线段；相应地，对于地理数据，应该使用类似 plotm(lat,long)的语法格式。

因为大部分地图中间都包含了几种信息，新的绘图命令向地图坐标系中添加图形元素而不是替换图形元素。另外，x 坐标和 y 坐标的比例设置为相等。用 daspectm 命令控制 z 轴的比例。

下面的例子显示如何通过在地图上单击找到点的地理位置，然后显示结果。将数据显示为没有标记的直线段，或显示为没有直线段的标记，改变线型的指定。

```
worldmap jamaica
[lat,lon] = inputm(4);
plotm(lat,lon,'-+r')
hidem(gca)
```

生成图36-9。

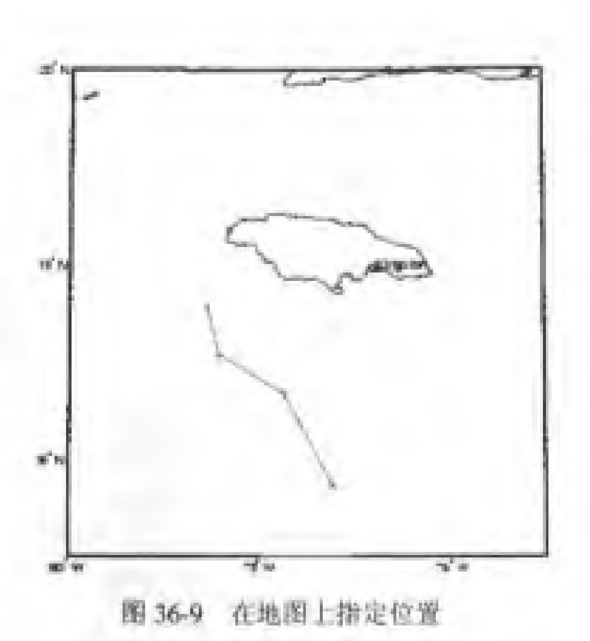


图36-9 在地图上指定位置

重要的地理线段数据如大圆跟踪线和小圆也可以进行交互计算和创建。大圆跟踪线对应于点间的最短距离，小圆是距离中心点固定距离的点的轨迹。

下面的例子绘制从美国到日本的大圆跟踪线和中心在东京，半径为1000 km的小圆。在地球上，跟踪线应该是直的，小圆是环形的。地图投影会导致某种变形。

```
axesm('ortho','origin',[30 180])
framem;gridm
plotm(coast,'k')
trackg
scircleg
```

生成图36-10。

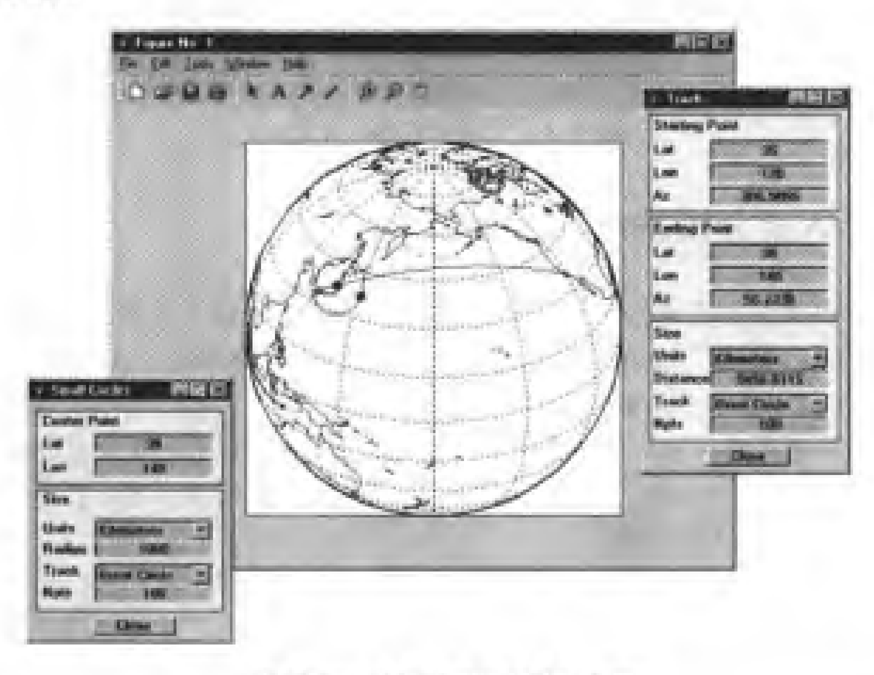


图36-10 添加大圆跟踪线和小圆

显示矩阵数据的一个通用办法是显示为等值线图。地图制作工具箱提供了多种方法来表示等值线。等值线用contourm命令创建，填充的等值线用contourfm命令创建。将矩阵显

示为表面并用 contourcmap 命令控制颜色查找表可以达到使用 contourfm 命令绘制等间隔的等值线的相同效果。

下面仍旧使用图 36-5 用过的朝鲜半岛的数据 geoid。controurm 和 clabelm 命令在地图上每隔 10m 显示和标注 geoid 高程的等值线。

```
load korea
worldmap(map,maplegend,'dem')
load geoid
[c,h] = contourm(geoid,geoidlegend,-100:10:100,'r');
ht = clabelm(c,h); set(ht,'color','r')
hidem(gca)
```

生成图 36-11。

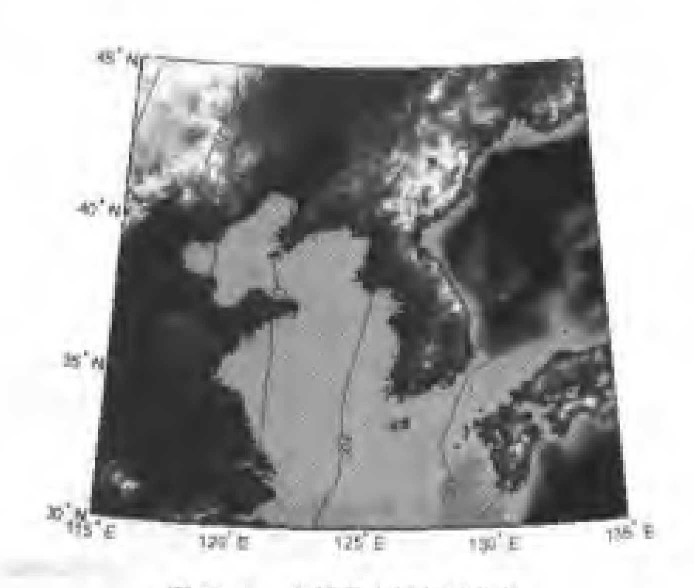


图 36-11　在地图上添加等值线

36.3　导入高分辨率地图集数据

除了工具箱提供的地图集数据外，还可以导入很多高分辨率数据集。这些数据集的水平分辨率为 10km～100m。支持的向量数据包括详细的向量海岸线、世界数字图形和它的继承者 VMAP0。可以用命令行函数和图形用户界面导入数据。

可以获得的分辨率最高的全球向量数据是 Vector Map Level0，或者 VMAP0。下面的例子演示如何启动 vmapOui 面板。然后可以缩放地图并从列表中选择图层。然后单击“Get”按钮，为目标区域提取选定的图层。

```
vmap0ui
```

生成图 36-12。

提取出所有需要的数据以后，可以将结果保存到一个 MAT 文件中，或者将它返回到工作空间中，以便在其他地图坐标系中进行显示。数据保存为地理数据结构，它包含了足够的信息使数据可以自动显示。图 36-13 中，Cape Cod 的所有地理数据都被提取出来，并保存在 MAT 文件 vmap0cape 中。worldmap 命令创建一个空白的底图，mlayers 面板用于交互式

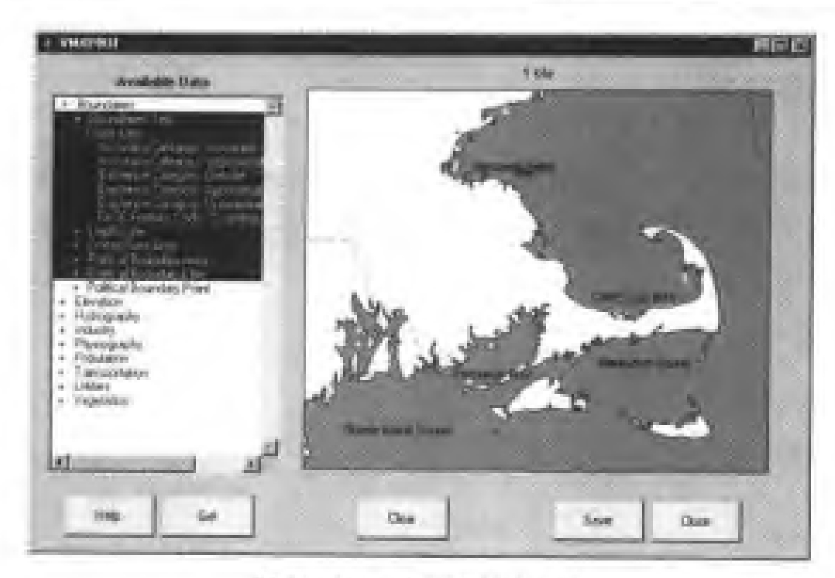


图 36-12　导入高分辨率数据

地逐层显示数据。也可以用命令行进行等价操作，需要载入 vmap0cape 数据并使用 displaym 命令。

```
figure
worldmap([41.2 42.2],[-71 -69.9],'none')
hidem(gca)
set(gcf,'color','w')
mlayers vmap0cape
```

生成图 36-13。

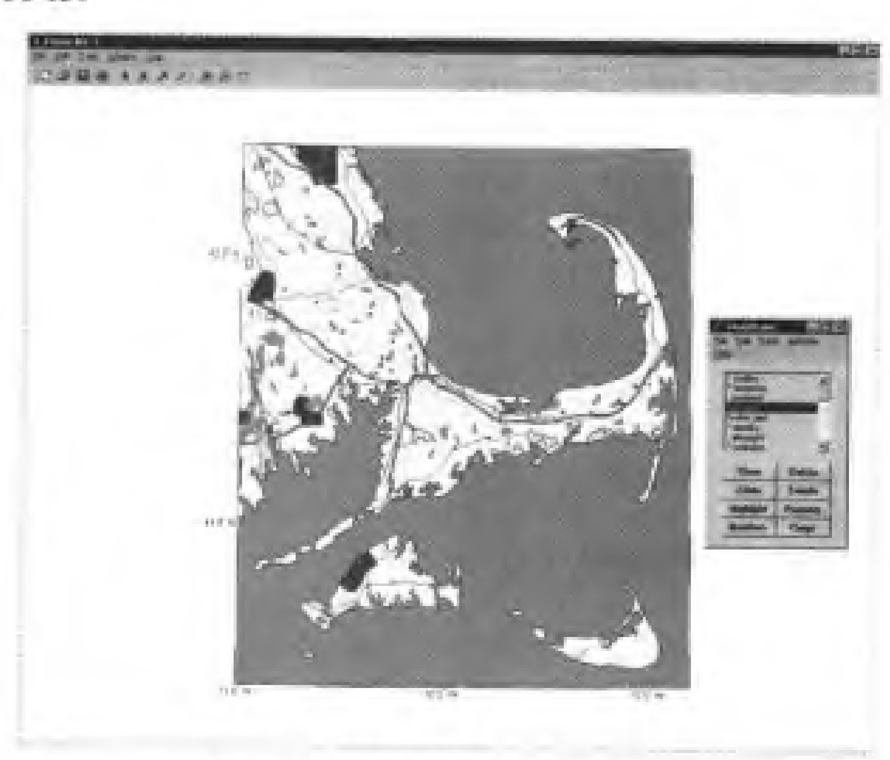


图 36-13　显示 Cape Cod 数据集中的地理数据

36.4 地理计算

除了地理显示能力外，工具箱还包括一个用于分析地理数据的扩展命令集。包括单位转换、地理几何如距离、方位、高程、定位和相交等方面的命令。也可以创建地理数据如跟踪线、圆、椭圆和缓冲区等。

```
nm2km(1)
ans =
        1.852
sm2km(1)
ans =
       1.6093
dst = distance(45,115,30,135)
dst =
       21.721
deg2km(dst)
ans =
       2415.3
azimuth(45,115,30,135)
ans =
        126.84
```

正则矩阵地图可以用于多种操作。矩阵地图能力包括通过地理定位、将向量转换为矩阵数据、编码闭合区域和计算坡度、方位和梯度，以及地形的可见性来插入和提取值。例如，下面计算黄海中某点的水深和 geoid 高程。

```
load korea
ltln2val(map,maplegend,35,125)
ans =
   -64
load geoid
ltln2val(geoid,geoidlegend,35,125)
ans =
                        19.7490
In which country is this location?
load worldmtxmed
code = ltln2val(map,maplegend,36,127)
code =
   104
names{code}
ans =
Korea, Republic of
```

第 37 章　地理空间数据

地图可以简单地定义为地理数据的图形表示。现在大部分人认为地图是二维的，但是，对于古埃及人来说，地图最初的形式是路线上先后要经过的地点的名称列表。现在，这个列表会被认为是地图数据，而不是地图。

地图制作工具箱中，地图数据是表示地理位置、区域属性或星球表面特征的任何变量或变量集，数据的大小、复杂度和格式没有限制。使用工具箱提供的函数和图形用户界面，可以将这些数据通过多种方法渲染成地图。

地理空间数据取自多种形式和格式的数据，它的结构比表或非地理测量数据更复杂。实际上，它是空间数据的一个子集，只表示点在给定坐标系中的位置。

37.1　地图数据

目前用于表示地形的地理空间数据类型主要有向量数据和栅格数据两种。向量数据用点、直线和多边形表示对象的形状，而栅格数据把空间分割成用值表示的单元。可以将这两种数据组合起来使用，在栅格数据上叠加向量数据进行显示。

37.1.1　向量数据

向量数据可以表示地图。这些向量用经度-纬度或投影坐标对表示点集、线形地图或区域地图的特征。使用这种表示方法时，地理数据以向量格式保存，绘成的地图称为向量地图。这些数据由指定的坐标点列组成，并用其他数据表示与相邻点的连接关系。

地图制作工具箱中，向量数据由先后连接的地理或投影坐标对组成，数据的分隔可以用分隔符 NaNs 表示，也可以创建单独的向量变量。对于向量地图数据，数据的连通性通常只影响图形显示，但它对于地图的统计计算也有影响。

下面结合一个例子介绍向量数据。

（1）在命令窗口键入下面的命令，载入数据集 coast。

```
load coast
whos
Name        Size                 Bytes   Class
lat       9589x1                 76712   double array
long      9589x1                 76712   double array
```

变量 lat 和 long 都是 coast 文件中的向量，它们一起组成了一幅世界海岸线向量地图。

（2）要查看该数据的地图，输入

```
axesm mercator
framem
```

```
plotm(lat,long)
```

生成的地图如图 37-1 所示。

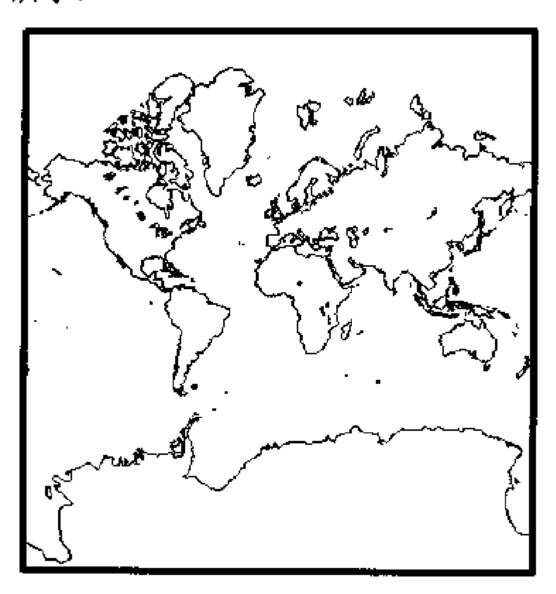


图 37-1　数据的地图显示

下面查看海岸线向量数据的前 20 个坐标。

```
[lat(1:20) long(1:20)]
ans =
   -83.8300 -180.0000
   -84.3300 -178.0000
   -84.5000 -174.0000
   -84.6700 -170.0000
   -84.9200 -166.0000
   -85.4200 -163.0000
   -85.4200 -158.0000
   -85.5800 -152.0000
   -85.3300 -146.0000
   -84.8300 -147.0000
   -84.5000 -151.0000
   -84.0000 -153.5000
   -83.5000 -153.0000
   -83.0000 -154.0000
   -82.5000 -154.0000
   -82.0000 -154.0000
   -81.5000 -154.5000
   -81.1700 -153.0000
   -81.0000 -150.0000
   -80.9200 -146.5000
```

（3）为了查看这些向量点表示的海岸线，键入下面的命令，用红色显示它们。

```
plotm(lat(1:20), long(1:20),'r')
```

上面的例子使用的是 Mercator 投影。通过投影，可以将球形表面用二维表面表示。有很多种投影方法，每种方法都会产生不同类型的变形。

37.1.2　栅格数栅

还可以将矩阵数据用地图表示，这个矩阵中每个行与列交叉点上的元素对应于特定地理区域的矩形面片，另有数据表示它与相邻面片的连通性。这通常称为栅格数据。当栅格格式的数据表示星球表面时，称为数据网格，数据保存为数组或矩阵。MATLAB 的矩阵操作函数完全适用于这类数据。栅格的值可用单元内的平均值或中心值表示。

当栅格数据由表面高程组成时，地图可称为数字高程模型（DEM），它显示为地形图。DEM 是数字地形模型（DTM）最通用的形式，还可以用等值线、三角高程点、四叉树、八叉树等表示。工具箱中的 topo 世界地形数据就是 DEM 的一个例子。在这个 180360 的矩阵中，每一行表示纬度的 1°，每一列表示经度的 1°。矩阵中的每个元素值为平均高程，单位为米。

栅格数据还包括地理引用图像数据。与数据网格类似，图像数据用行和列表示。但它们之间有一些小小的差别，理解这些差别很重要。差别之一是，图像可能将 RGB 值或多谱信道（multispectral channel）保存在一个单独的数组中，这样它就有了第 3 个维。另一个差别是，当数据网格保存为 double 型时，图像可能使用 MATLAB 保存类型的数据范围，最通用的是 uint8、uint16、double 和 logical 等。第 3 个差别是，对于 double 型的灰度图像或 RGB 图像，数组元素中的值限制在[0 1]内部。

下面结合一个例子介绍模格数据。

（1）输入下面的命令，载入 topo 数据集并显示它的结构。

```
clear all;
load topo
whos
  Name          Size           Bytes  Class

  topo          180x360       518400  double array
  topolegend    1x3               24  double array
  topomap1      64x3            1536  double array
  topomap2      128x3           3072  double array

Grand total is 65379 elements using 523032 bytes
```

在工作空间面板中双击 topo 变量名或在命令窗口输入下面的命令，用 MATLAB 数组编辑器查看 topo 变量中的栅格高程数据。

```
openvar topo
```

（2）可以看出，topo 似乎是一个二维数组，并且它的左上角附近的值界于 2500 和 3000 之间。第 1 行表示南极附近的陆地高程。

（3）下面创建一个等面积地图投影，查看地形数据。

```
axesm sinusoid
```

显示一个图形窗口，设置地图坐标系显示该投影。

（4）生成一个阴影地貌图。可以通过多种方法实现，首先用 geoshow 函数进行显示，然后用 demcmap 函数应用一个用于地形显示的颜色查找表。

```
geoshow(topo,topolegend,'DisplayType','texturemap')
demcmap(topo)
```

geoshow 函数将 geodata 数据显示在地理坐标系中。输出如图 37-2 所示。

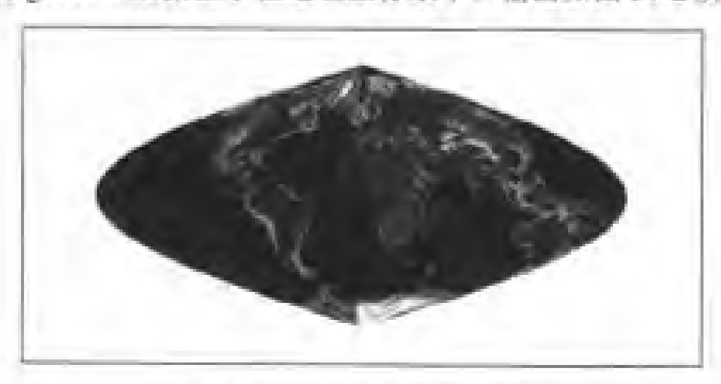


图 37-2　geodata 数据的地图显示

（5）现在用 Hammer 投影创建一幅新图像，并用 meshlsrm 函数显示 topo 数据，该函数允许控制光照效果。

```
figure; axesm hammer
meshlsrm(topo,topolegend)
```

现在，同样是 topo 数据集生成的着色后的地貌图，从东方加光照以后得到的渲染效果显示在第 2 个图形窗口中，如图 37-3 所示。

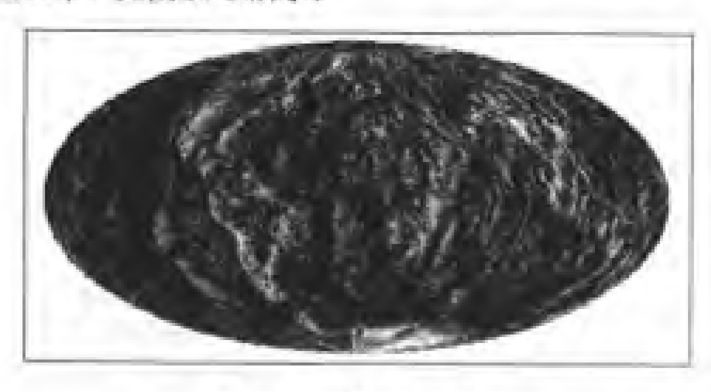


图 37-3　着色后的地貌图

向量地图变量和数据网格变量常常一起使用或显示。例如，向量形式的大陆架可能与一个温度数据网格一起使用，并使后者更有用。当多个地图变量不管变量类型一起使用时，可以认为是一套单独的地图数据。当然，要做到这一点，不同的数据集必须使用相同的坐标系。

使用前面例子中使用过的 coast 和 topo 数据集，把它们组合到一张地图中，并观察两种类型的数据是如何一起工作的。

（1）清除当前地图。

```
clma
```

（2）重新载入海岸线数据。

```
load coast
```

（3）如果 topo 数据在工作空间中不存在，也载入它。

```
load topo
```

（4）设置 Robinson 投影。

```
axesm robinson
```

（5）用合适的颜色查找表绘制栅格地形数据的图。

```
geoshow(topo,topolegend,'DisplayType','texturemap')
demcmap(topo)
```

（6）在地形图上方绘制海岸线数据的图。

```
geoshow(lat,long,'Color','r')
```

注意，可以用 geoshow 函数同时显示栅格数据和向量数据。生成的地图如图 37-4 所示。

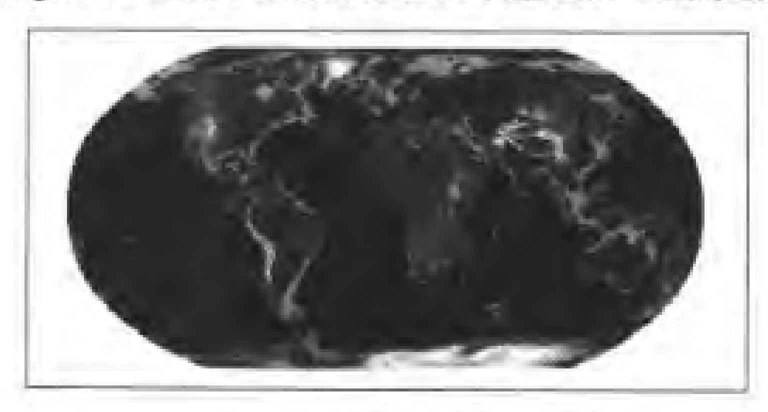


图 37-4　同时显示栅格数据和向量数据

37.2　操作向量数据

使用地图制作工具箱可以用不同的方式操作、组合和区分向量地理数据。下面介绍一些有助于选择和转换向量地理数据的函数。

37.2.1　重新组装向量对象

当部分直线和面片被组合到元素用 NaN 分隔的很大的向量中以后很难识别。可以用 polysplit 函数把这些多边形或直线向量分解成基本图形单元，该函数将列向量作为输入变量。

下面的例子用 polysplit 函数提取和连接多边形或直线段。

（1）输入两个列向量形式的以 NaN 为界的数组。

```
lat = [45.6 -23.47 78 NaN 43.9 -67.14 90 -89]';
long = [13 -97.45 165 NaN 0 -114.2 -18 0]';
```

（2）用 polysplit 函数创建两个单元数组 latc 和 lonc。

```
[latc,lonc] = polysplit(lat,long)
```

```
latc =
    [3x1 double]    [4x1 double]
lonc =
    [3x1 double]    [4x1 double]
```

（3）查看单元数组的内容。

```
[latc{1} lonc{1}]
ans =
        45.6          13
      -23.47      -97.45
          78         165
[latc{2} lonc{2}]
ans =
        43.9           0
      -67.14      -114.2
          90         -18
         -89           0
```

注意，每个单元数组元素包含原直线的一段。

（4）进行反操作，使用 polyjoin 函数。

```
[lat2,lon2] = polyjoin(latc,lonc);
```

（5）连接后的线段与最初的 lat 和 lon 数组等价。

```
[lat long] == [lat2 lon2]
ans =
    1    1
    1    1
    1    1
    0    0
    1    1
    1    1
    1    1
    1    1
```

（6）可以测试两套数据是否完全相等，键入下面的命令行。

```
isequalwithequalnans(lat,lat2) & isequalwithequalnans(long,lon2)
ans =
    1
```

37.2.2 匹配直线段

将具有公共端点的直线段进行连接是与直线段有关的一个通用操作。polymerge 命令比较保存在纬度数据向量和经度数据向量中的线段端点数据，以便识别精确匹配或距离在一定范围内的端点，然后将匹配的直线段进行连接。重复此过程，直到没有公用点可以找到为止。有两个必需的变量是纬度向量和经度向量。下面的例子通过将直线段连接成多边形演示

了这个过程。

（1）创建表示坐标值的列向量。

```
lat = [3 2 NaN 1 2 NaN 5 6 NaN 3 4]';
lon = [13 12 NaN 11 12 NaN 15 16 NaN 13 14]';
```

（2）将精确匹配的线段连接起来。

```
[latm,lonm] = polymerge(lat,lon)
ans =
     5    15
     6    16
   NaN   NaN
     1    11
     2    12
     2    12
     3    13
     3    13
     4    14
```

原来的 4 条线段变成了两条。

polymerge 函数还有可选的第 3 个变量，它设置允许不精确匹配的距离容限。第 4 个变量使用户可以指定函数输出向量还是单元数组。

37.2.3　地理插值

使用向量数据，作与点间地理真实性有关的假设时必须小心。例如，用向量数据绘图时，可能会用直线段连接每个点。通常这并不表示这些点之间的任何真实信息。沿海岸线的点的分布可能比较稀疏，在缺少其他信息的情况下，在两点之间插入其他点可能会导致错误，如图 37-5 所示。

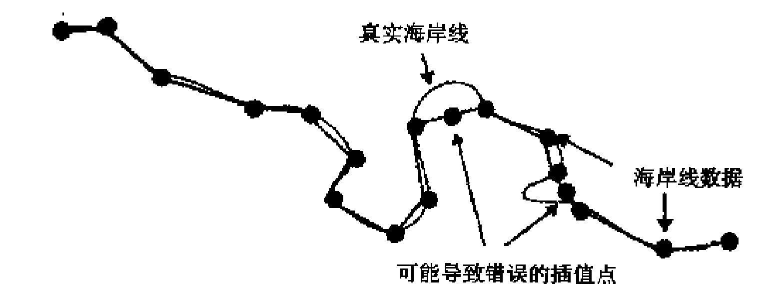


图 37-5　地理数据插值

尽管进行地理插值会有一定风险，但在很多情况下它还是有用或者必要的。更稀疏的数指可以用 interpm 函数进行线性填充。

考虑一系列纬度和经度点，对它们进行线性插值处理，使得在每个方向上的距离都不超过 1°。

```
lats = [1 2 4 5]; longs = [1 3 4 5]; maxdiff = 1;
[newlats,newlongs] = interpm(lats,longs,maxdiff)
```

```
newlats =
     1.0000
     1.5000
     2.0000
     3.0000
     4.0000
     5.0000
newlongs =
     1.0000
     2.0000
     3.0000
     3.5000
     4.0000
     5.0000
```

在原来的 lats 数据中，2 和 4 之间相差 2，所以，newlats 和 newlongs 两个新变量中插入了（3, 3.5）这个点。类似地，原来的 longs 数据中，1 和 3 之间也相差 2，所以 newlats 和 newlongs 两个变量中插入了（1.5, 2）这个点。现在，newlats 和 newlongs 中再没有相邻的点的距离比 maxdiff 还大了。

interpm 函数在原数据中插入新插值点以后返回。但是，有时候只需要插值点，命令 intrplat 和 intrplon 提供了与 MATLAB 命令 interp1 命令相似的功能，可以用不同方法进行插值。

用 intrplat 函数进行插值，可以选择用线性、样条、三次、恒向线或大圆弧等方法进行插值。利用下面给定的数据进行计算，求与经度 7.3° 对应的纬度，用线性、大圆和恒向线 3 种方法进行计算。

```
longs = [1 3 4 9 13]; lats = [57 68 60 65 56]; newlong = 7.3;
newlat = intrplat(longs,lats,newlong,'linear')
newlat =
     63.3000
newlat = intrplat(longs,lats,newlong,'gc')
newlat =
     63.5029
newlat = intrplat(longs,lats,newlong,'rh')
newlat =
     63.3937
```

intrplon 函数的功能与 intrplat 的相似，只不过它求的是经度。

37.2.4 向量相交

地图制作工具箱提供了一系列函数对向量数据进行求交计算。这些函数还可以计算任意向量数据的交。

下面的例子计算圆心在（0°, 0°），半径为 1250 海里的小圆与圆心在（5°N, 30°E)，半

径为 2500 公里的小圆的交点。

```
[lat,long] = scxsc(0,0,nm2deg(1250),5,30,km2deg(2500))
lat =
     17.7487 -12.9839
long =
     11.0624 16.4170
```

结果如图 37-6 所示。

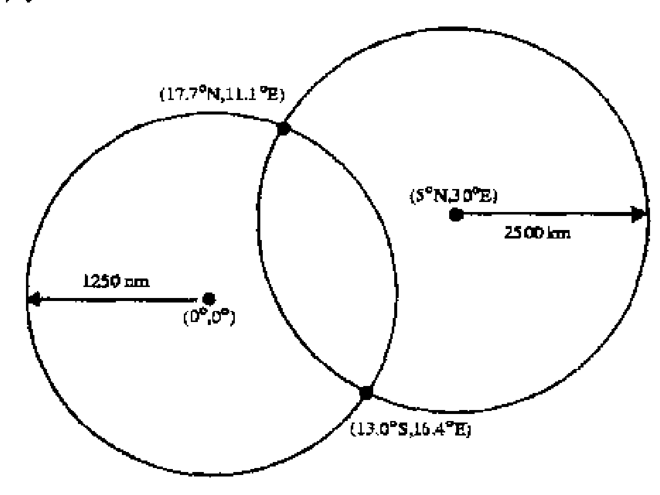


图 37-6　两个相交的小圆

其他功能相似的命令包括 rhxrh、gcxgc 和 gcxsc 等，其中，rhxrh 函数计算恒向线的交点，gcxgc 函数计算大圆的交点，gcxsc 函数计算大圆和小圆的交点。

37.2.5　多边形的面积

可以用函数 areaint 计算多边形格式向量数据的地理面积。下面用 usalo 数据集计算美国大陆的面积。

```
load usalo
earthradius = almanac('earth','radius');
area = areaint(uslat,uslon,earthradius)
area =
1.0e+06 *
          7.9256
          0.0035
          0.0004
```

37.2.6　通过布尔操作叠加多边形

多边形布尔操作可以回答一系列与向量数据多边形对象逻辑关系有关的问题。标准布尔操作包括交、并、差和异或等。polybool 函数可以对两个向量集合进行这些操作。

下面举一个例子演示 polybool 函数的使用。

（1）创建一个十二边形。

```
theta = (0:pi/6:2*pi)';
lat1 = sin(theta);
lon1 = cos(theta);
```

（2）创建一个三角形叠加到十二边形上。

```
lat2 = [0 1 -1 0]';
lon2 = [0 2 2 0]';
```

（3）用蓝线和红线将这两个形状显示在一起。

```
axesm miller
plotm(lat1,lon1,'b')
plotm(lat2,lon2,'r')
```

（4）计算多边形的交，并把它绘成绿色面片。

```
[lati,loni] = polybool('intersection',lat1,lon1,lat2,lon2);
[lati loni]
ans =
       0.44093      0.88185
   1.2246e-016            1
      -0.44093      0.88185
   1.2246e-016  6.1232e-017
       0.44093      0.88185
patchm(lati,loni,'g')
```

（5）计算多边形的并，并绘成洋红色面片。

```
[latu,lonu] = polybool('union',lat1,lon1,lat2,lon2);
[latu lonu]
ans =
       0.44093      0.88185
             1            2
            -1            2
      -0.44093      0.88185
          -0.5      0.86603
      -0.86603          0.5
            -1  6.1232e-017
      -0.86603         -0.5
          -0.5     -0.86603
   1.2246e-016           -1
           0.5     -0.86603
       0.86603         -0.5
             1  6.1232e-017
       0.86603          0.5
           0.5      0.86603
       0.44093      0.88185
patchm(latu,lonu,'m')
```

（6）对多边形进行异或计算，并把结果显示成黄色面片。

```
[latx,lonx] = polybool('xor',lat1,lon1,lat2,lon2);
[latx lonx]
ans =
      -0.44093      0.88185
    1.2246e-016           1
       0.44093      0.88185
             1            2
            -1            2
      -0.44093      0.88185
           NaN          NaN
       0.44093      0.88185
    1.2246e-016  6.1232e-017
      -0.44093      0.88185
          -0.5      0.86603
      -0.86603          0.5
            -1  6.1232e-017
      -0.86603         -0.5
          -0.5     -0.86603
    1.2246e-016          -1
           0.5     -0.86603
       0.86603         -0.5
             1  6.1232e-017
       0.86603          0.5
           0.5      0.86603
       0.44093      0.88185
patchm(latx,lonx,'y')
```

（7）最后，求十二边形与三角形的差，将生成的凹多边形绘成白色面片。

```
[latm,lonm] = polybool('minus',lat1,lon1,lat2,lon2);
[latm lonm]
ans =
       0.44093      0.88185
    1.2246e-016  6.1232e-017
      -0.44093      0.88185
          -0.5      0.86603
      -0.86603          0.5
            -1  6.1232e-017
      -0.86603         -0.5
          -0.5     -0.86603
    1.2246e-016          -1
           0.5     -0.86603
```

```
      0.86603         -0.5
            1  6.1232e-017
      0.86603          0.5
          0.5      0.86603
      0.44093      0.88185
patchm(latm,lonm,'w')
```

最终的结果如图 37-7 所示。

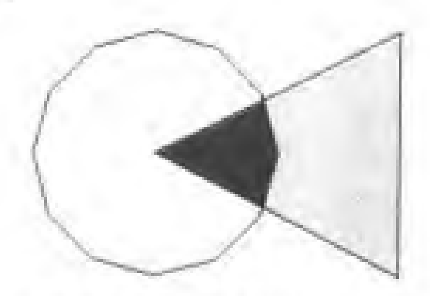


图 37-7　多边形的布尔运算

37.2.7　生成缓冲区

缓冲区是地图中地物周围的一个区域，这个区域中的点与地物的最大距离是指定了的。对于栅格地理数据，缓冲区是一个连续的编码相同的网格单元集。当地物呈多边形时，缓冲区可以定义为距离其边界一定距离的点的轨迹与地物多边形围成的区域。缓冲区边界就像是围绕对象的等值线。

bufferm 函数计算和返回一个向量，该向量代表定义缓冲区的点集。下面结合意大利地图演示 bufferm 函数的使用。计算距离意大利国界外 1.5° 的缓冲区。

（1）创建一个包围意大利的区域的基准图，隐藏它的边界。

```
close all; clear all;
worldmap('lo', [35,50], [3,23], 'lineonly')
hidem(gca)
```

（2）读入 worldlo 数据并提取出表示意大利的多边形。

```
load worldlo
[ilat,ilon] = extractm(POpatch,'italy');
```

（3）用 bufferm 函数处理意大利多边形，输出一个距离国界 1.5° 的缓冲区。

```
[latb,lonb] = bufferm(ilat,ilon,1.5,'out');
```

输出过程需要几分钟的时间，因为计算缓冲区需要比较大的计算量。

（4）把缓冲区绘成黄色，把意大利界内绘成绿色。

```
patchesm(latb, lonb, 'y')
patchesm(ilat, ilon, 'g')
```

显示效果如图 37-8 所示。

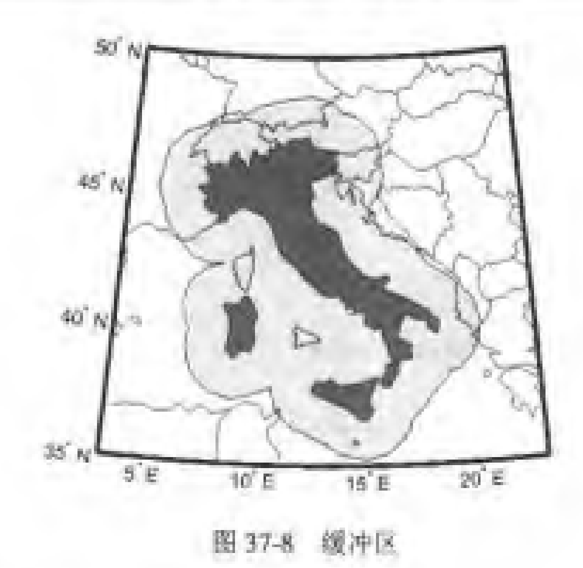


图37-8 缓冲区

37.3 操作栅格数据

有一些操作如属性的逻辑操作、表面特性的计算等使用栅格数据更合适。下面介绍几种操作栅格数据的方法。

37.3.1 向量数据和栅格数据的转换

可以将纬度-经度向量数据转换为任意精度的网格数据，然后可以使用该网格数据制作栅格图。地图制作工具箱提供了进行此类转换的图形用户界面，也可以从命令行中进行转换。向量数据网格化的主要函数是 vec2mtx。如果向量数据由多边形组成，则网格化轮廓都是空的，可以用 encodem 函数离散它们。

下面的例子根据向量数据创建数据网格。为了演示向量数据和栅格数据的转换，从 worldlo 数据集中提取出瑞士的面片数据。

（1）用 extractm 函数获取瑞士国界的面片数据。

```
close all; clear all;
[swlat,swlon] = extractm(worldlo('POpatch'),'Switzerland');
```

（2）将网格密度设置为 40 单元/度，用 vec2mtx 函数光栅化国界，并生成它的引用向量。

```
den = 40;
[swgrid,swrv] = vec2mtx(swlat,swlon,den);
whos
  Name        Size                 Bytes  Class
  den          1x1                     8  double array
  swgrid      80x186              119040  double array
  swlat       81x1                   648  double array
  swlon       81x1                   648  double array
  swrv         1x3                    24  double array

Grand total is 15046 elements using 120368 bytes
```

生成的网格数据是一个 double 型数组，有 80 行，186 列。

（3）用对比色绘制数据网格的地图。

```
axesm eqdcyl
meshm(swgrid,swrv)
colormap jet(4)
```

（4）设置地图的范围限制。

```
[latlim lonlim]=limitm(swgrid,swrv);
setm(gca, 'Flatlimit', latlim, 'FlonLimit', lonlim)
tightmap
```

（5）为了填充瑞士的内部区域，需要一个种子点和一个种子值。选择网格中间的行和列，并在调用 encodem 函数生成新网格时选择索引值 3 来识别瑞士的版图。

```
swpt = [size(swgrid)/2, 3]
swpt =
    40    93    3
swgrid3 = encodem(swgrid,swpt,1)
```

最后一个变量识别边界单元的编码。

（6）用填充后的网格清除和重绘地图。

```
meshm(swgrid3,swrv)
```

（7）在网格上绘原向量，比较栅格化的效果。

```
plotm(swlat,swlon,'k')
```

生成的地图如图 37-9 所示。



图 37-9　转换数据类型

37.3.2　用 GUI 光栅化多边形

可以把用 getseeds GUI 获得的种子点数据作为参数传递给 seedm 函数，用它填充网格化后的多个多边形。

（1）将一个包含有瑞士及其邻国名称的单元数组传递给 extractm 函数，提取他们的版图数据。

```
close all; clear all;
pcs={'Switzerland','Germany','Austria', 'Italy', 'France'};
[celat,celon] = extractm(worldlo('POpatch'),pcs);
```

（2）用 maptrimp 函数将国界裁剪到指定范围。

```
[polylat,polylon] = maptrimp(swlat,swlon,[45 49],[5 11]);
```

（3）放大裁剪后多边形的大小至 60 单元/度的分辨率同样会生成一个引用向量。

```
[cegrid,cerv] = vec2mtx(polylat,polylon,60);
```

（4）新建一个地图图形窗口，显示刚创建的网格。

```
axesm eqdcyl
meshm(cegrid,cerv)
colormap jet(8)
```

（5）用 getseeds 函数交互式地选择瑞士及其邻国的种子点。

```
[row,col,val] = getseeds(cegrid,cerv,5,[3 4 5 6 7])
row =
    119
    197
    136
     40
     49
col =
    177
    240
    339
    303
     33
val =
      3
      4
      5
      6
      7
```

因为这是交互式进行的，行号和列号可以不同。

（6）用 encodem 函数单色填充每个国家的版图，生成一个新的网格 cegrid5。

```
cegrid5 = encodem(cegrid,[row col val],1);
```

（7）清除显示结果并显示网格 cegrid5。

```
clma
meshm(cegrid5,cerv)
```

改变大小后的瑞士及其邻国的版图如图 37-10 所示。

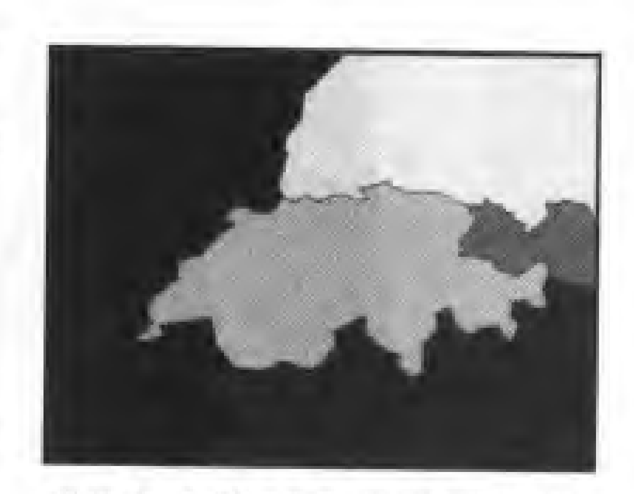


图 37-10　改变的大小后的瑞士及其邻国版图

37.3.3　路径上的数据网格值

网格化地理数据的一个通用应用是计算路径上的数据值，例如，计算一个横剖面、道路或飞行路线上的地形线高程等。可以用 mapprofile 函数完成此项计算。

下面的例子用 mapprofile 函数计算直线上的高程。

（1）载入韩国的地形高程数据。

```
clear all; close all;
load korea
```

（2）用 limitm 函数获取纬度和经度范围限制，然后利用其结果，通过 worldmap 函数生成图框。

```
[latlim,lonlim] = limitm(map,maplegend);
worldmap(latlim,lonlim,'none')
```

（3）给地图着色并给它应用数字高程模型（DEM）颜色查找表。

```
meshm(map,maplegend,size(map),map)
demcmap(map)
```

（4）定义穿过区域的直线段的端点。

```
plat = [40.5 37.7];
plon = [121.5 133.5];
```

（5）现在计算高程，跟踪类型为大圆，插值类型为双线性。

```
[z,rng,lat,lon] = mapprofile(map,maplegend,plat,plon);
```

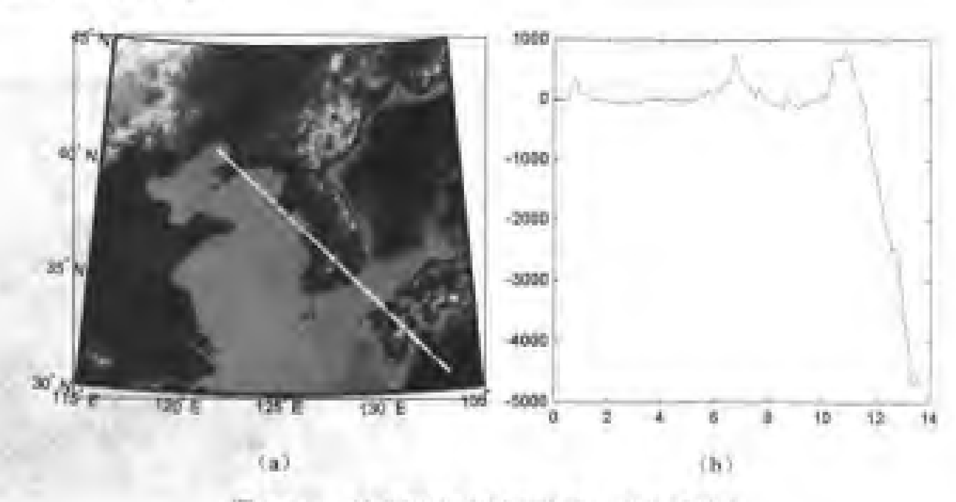
（6）绘制沿地形的三维横剖面。

```
plot3m(lat,lon,z,'w','LineWidth',2)
```

（7）生成横剖面对应的地形线。

```
figure; plot(rng,z,'r')
```

结果如图 37-11 所示。



（a）　（b）

图 37-11　地形图上的剖面线及对应的地形线

第 38 章　地理空间几何

本章介绍一些地理空间几何方面的基础知识，包括地球体、纬度、经度、大圆、恒向线和小圆等，这些概念理解好对于后续知识的理解很有用。

38.1　球体、椭球体和地球体

尽管地球很圆，但它实际上是一个椭球体，而不是精确的球体。这个差别很小，小到只有地球大小的 1/300，以至于制作小比例尺地图时可以把地球看作球体。但是，制作精度更高的大比例尺地图时需要使用椭球体模型。这个模型是必需的，例如用高分辨率卫星图像或航空图像制作地图时，或者使用 GPS 坐标数据进行操作时，都需要这个模型。下面介绍地图制作工具箱如何精确建立地球和其他星球的形状或图像模型。

38.1.1　地球体和椭球体

地球体是地球图像的经验近似。它是一个与重力有关的等位面，或多或少都与海平面相对应。它可以近似地看作扁平的椭球体，但还不完全如此，因为地表还有一些局部的起伏。

1．绘制地球体

用 geoid 数据集绘制地球的图像，在命令窗口输入

```
clear;
load geoid; load coast
figure; axesm robinson
meshm(geoid,geoidlegend)
colorbar('horiz')
plotm(lat,long,'k')
```

生成图 38-1。

地球的形状对于某些应用来说很重要，比如计算卫星的运行轨道，当然也不是对每种应用都重要。但有时了解地球体方面的知识是必要的，例如，比较海平面以上的高度与 GPS 测量得到的高度时就需要。



图 38-1　世界地图

计算地理空间坐标时，地球体通常是作为椭球体处理的。定义椭球体有几种方法。通常用长半轴和短半轴进行定义，也常用长半轴和扁率或偏心率来描述。不管使用哪套

参数，都可以完全控制椭球体，并推导出其他参数。椭球体的组成要素如图 38-2 所示。

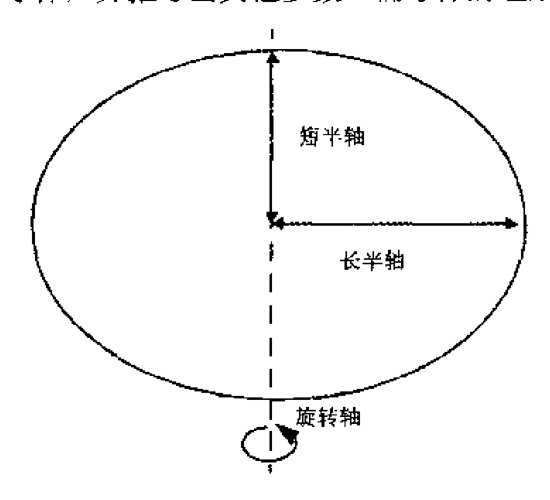


图 38-2　椭球体的组成要素

38.1.2　椭球体向量

工具箱中的椭球体通常用 2 元素向量表示，称为椭球体向量。椭球体向量具有[长半轴 偏心率]的形式。长半轴的长度表示可用任何长度单位，最常用的单位是米或公里。

偏心率可以界于 0 和 1 之间。只提供一个元素时，假设偏心率为 0，即为球体。默认时采用的椭球体是 1980 年测绘参考系统得到的椭球体。

```
almanac('earth','ellipsoid','kilometers')
ans =
   1.0e+03 *
   6.37813700000000     0.00008181919104
```

对比球形椭球体的定义：

```
almanac('earth','sphere','kilometers')
ans =
          6371          0
```

almanac 函数把关键字'geoid'作为'ellipsoid'看待。

工具箱中的 almanac 函数提供了多种星球椭球体向量的标准值和其他几种数据。例如，下面用 almanac 函数查看 wgs72 椭球体的参数。

```
wgs72 = almanac('earth','wgs72')
wgs72 =
6378.135          0.0818188106627487
```

将它与 Bessel 的 1841 年椭球体比较。

```
bessel = almanac('earth','bessel')
bessel =
                    6377.397155          0.0816968329674029
```

椭球体向量的元素值为长半轴长度和偏心率，长度单位为公里。偏心率和扁率都用于度量椭球体不同维的长度比率。工具箱中提供了多个函数将椭球体的这种表示形式转化为其他形式。例如，把长半轴和短半轴作为参数时，函数 axes2ecc 返回一个偏心率。

地球的长半轴比短半轴长 21 km，下面用 almanac 函数计算。

```
grs80 = almanac('earth','ellipsoid','kilometers')
grs80 =
            6378.137          0.0818191910428158
semiminor = minaxis(grs80)
semiminor =
         6356.75231414036
semidiff = grs80(1) - semiminor
semidiff =
        21.3846858596444
```

与长半轴 6 400 km 比较，这个差距很小，对于世界地图和其他比例尺更小的地图，可以忽略不计。例如，如果一幅地图上 21.38 km 的长度可以用小于 0.5 mm 的直线段表示，则它的比例尺为

```
nodiff = semidiff * 1e6 / 0.5
nodiff =
  4.2769e+007
```

因子 1e6 将距离 semidiff 简单地从公里转化为厘米。这表示地球的偏心率不能用小于 1∶40000000 的比例尺描述，它大致上是世界地图显示在这个页面上时的比例尺。所以，工具箱中的大部分函数默认时都将地球设置为球形模型。但可以指定其他任何模型。

38.2 纬度和经度

用纬度和经度可以指定星球表面上点的位置。

纬度是赤道平面与点和旋转轴连线的夹角。北半球纬度为正，北极纬度为 +90°；南半球纬度为负，南模纬度为 -90°。固定纬度上的线称为纬线。如图 38-3 所示。

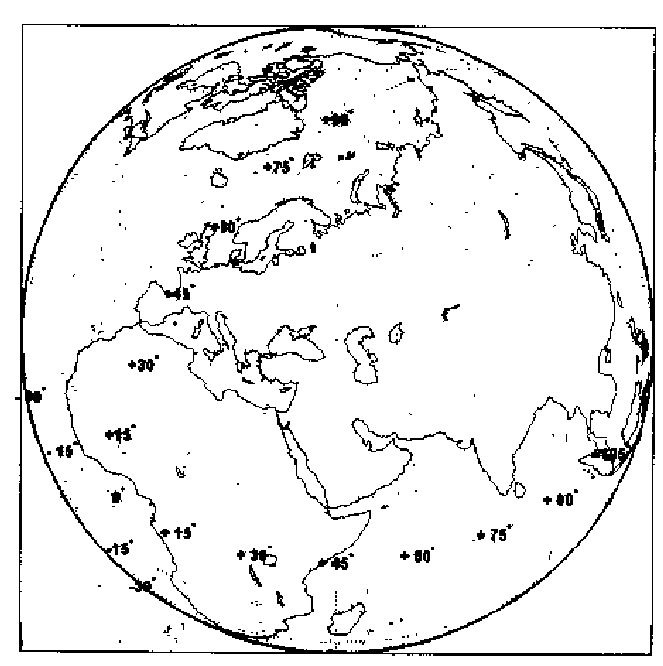


图 38-3　纬度和经度

经度是两个平面的夹角，其中一个平面是点和旋转轴确定的平面，另一个是本初子午面，即 0° 面。本初子午面用英国格林威治皇家天文台的位置确定。固定经度的线称为经线。所有经线都集中到北极和南极。

经度的变化范围通常为 -180° ～ +180°，但也有例外，比如从 0° ～ +360°。经度也可以用其他方法指定，比如从 0° ～180°E 和从 0° ～180°W。对经度增加或减小 360° 不改变点的位置。

38.3 大圆、恒向线和小圆

在平面几何中，直线具有两个重要的特性：

- 直线表示两点之间的最短路径；
- 直线的斜率是一定的。

但是，在球体的曲面上描述直线时，一次只能保证这两个特性中的一个有效。

38.3.1 大圆

大圆是球体曲面上两点之间的最短路径。大圆的精确定义是两点与球心确定的平面与球体曲面的交线。这样，大圆总是将球体对半分开。赤道和所有子午线都是大圆。大圆是点间的最短路径，在地图上并不总是显而易见，因为很少有地图投影将大圆表示成直线。

38.3.2 恒向线

恒向线是一根曲线，它沿固定角度穿过所有子午线。尽管大圆是最短路径，但很难用它导航，因为前进时方位角总是在改变。沿恒向线比沿测地线要走更长的路，但更容易确定方向。

所有纬线，包括赤道都是恒向线，因为它们穿过了所有子午线。另外，所有子午线除了是大圆外，还都是恒向线。除非恒向线的方向为正东、正西、正北或正南，它总是呈螺旋形向极点前进。

图 38-4 显示了一个大圆与一根恒向线的相交情况。

图 38-4　大圆与恒向线的相交情况

38.3.3 小圆

除了恒向线和大圆外，另一个重要的平滑曲线是小圆。一定纬度上的纬线都是小圆。小圆更一般的定义是：球面与水平表面的交线。在椭球体上，按照这个定义只有当定义的平面与赤道平行时才会生成真正的小圆。

定义小圆最简单的方法是用到点的距离进行定义。比如，所有距离（45°N, 60°E）45 海里的点都可以定义一个小圆。如果用弧长度数进行距离度量，则球面上大圆是所有距离特定中心点 90° 远的点集。

对于真正的小圆，距离必须在大圆意义上定义，即必须是球面上两点之间的最短距离。但是，工具箱还能计算斜向小圆，此时计算恒向线意义上的距离。

38.4　球体或椭球体上的角度和方向

方位角是直线与子午线所成的角度，从正北方向开始，按顺时针方向测量。这样，正北方向的方位角为 0°，正东方向为 90°，正南为 180°，正西为 270°。可以用地图制作工具箱计算任何成对点位置的方位角，或者沿恒向线，或者沿大圆。除了沿正方向外，两种方式会生成不同的结果。

沿恒向线时，反向计算方位角，即从第 2 个点向第 1 个点计算，生成前向方位角的余角。沿大圆时，反方位角通常不是余角，其差别与两点之间的距离有关。

除了前向方位角和反向方位角外，工具箱还可以计算相对于参考点给定了距离和方位角的点，也分沿大圆和沿恒向线两种情况。

38.4.1　定位——前向问题

地理应用中一个常见的问题是给定起点位置、初始方位角和距离时计算目标点的方位。工具箱中，这个过程称为定位。新位置可以有大圆和恒向线两种意义上的推算。

作为示例，假设一架飞机从纽约的 La Guardia 飞机场出发，沿西北方向的恒向线以 200 海里每小时的速度前进，求它一小时以后的位置。下面用 reckon 函数进行计算。

```
[rhlat,rhlong] = reckon('rh',40.75,-73.9,nm2deg(200),315)
rhlat =
    43.1054
rhlong =
    -77.0665
```

注意，距离 200 海里需要用 nm2deg 函数转换为表示弧长的度数，以与纬度和经度输入相匹配。如果飞机上有一台计算机，利用它可以沿一条精确的大圆路径飞行，一小时以后飞机的位置又在哪里呢？在命令窗口键入下面的命令行：

```
[gclat,gclong] = reckon('gc',40.75,-73.9,nm2deg(200),315)
gclat =
    43.0615
gclong =
    -77.1238
```

结果区别不大。

38.4.2　计算跟踪路径——大圆和恒向线

可以用 track1 或 track2 函数，对应于沿大圆或恒向线的点生成向量数据。如果在路径上有一个点和该点上的一个方位角，使用 track1 函数；如果路径上有两个点，使用 track2 函数。例如，起点为(31°S, 90°E)，方位角为 45°，距离为 12°，求大圆上的跟踪点，用 track1

函数计算：

```
[latgc,longc] = track1('gc',-31,90,45,12);
```

起点为(31°S, 90°E)，终点为(23°S, 110°E)，用 track2 函数计算：

```
[latgc,longc] = track2('gc',-31,90,-23,110);
```

track1 函数还允许在一定范围内指定端点的位置。例如，如果指定起点距离初始点 5°，终点距离初始点 13°，方位角为 55°，可以像下面这样指定限制范围。

```
[latrh,lonrh] = track1('rh',-31,90,55,[5 13]);
```

没有范围提供给 track1 函数时，返回点表示完整的跟踪路径。对于大圆的情况，完整路径是 360°，包围星球并返回初始点；对于恒向线的情况，除非方位角为 90° 或 270°，完整的跟踪在极点处终止，此时完整路径为返回到初始点的纬线。

38.4.3 距离、方位角和反方位角（反向问题）

用工具箱计算两点之间的距离时，结果取决于想要的是大圆距离还是恒向线距离。distance 函数用弧长角度返回两点之间的合适距离，对输入纬度和经度采用相同的角度单位。默认的路径类型是更短的大圆，默认的角度单位是度。点(15°S, 0°)和点(60°N, 150°E)之间的大圆距离，用弧度表示为

```
distgc = distance(-15,0,60,150)
distgc =
    129.9712
```

恒向线距离更大

```
distrh = distance('rh',-15,0,60,150)
distrh =
    145.0288
```

要确定恒向线路径长多少，并用公里表示，可以用距离转换函数进行计算。

```
kmdifference = deg2km(distrh-distgc)
kmdifference =
    1.6744e+03
```

工具箱中有几个转换函数可以使用，它们支持度、弧度、公里、米、英里、海里和英尺等。在弧长角度单位和曲面长度单位之间转换需要星球或椭球体的半径。默认时使用地球的半径。

38.4.4 计算方位角和仰角

方位角是直线与子午线所成的角度，从正北方向开始，按顺时针方向测量。用工具箱计算点至另一点的方位角时，结果跟选择大圆还是恒向线有关。对于大圆的情况，得到的方位角是大圆起点的方位角，通常，沿大圆的方位角不是常数。对于恒向线的情况，生成的方位角沿整个路径都是常数。

方位角的单位与输入的纬度和经度的相同。默认的路径类型是大圆，单位为度。下面的例子中，第 1 点到第 2 点的大圆方位角为

```
azgc = azimuth(-15,0,60,150)
azgc =
    19.0391
```

对于恒向线的情况，保持恒定值的方位角：

```
azrh = azimuth('rh',-15,0,60,150)
azrh =
    58.8595
```

恒向线的一个特性是，反方位角是前向方位角的补角，可以通过简单地给前向值添加 180° 进行计算。

```
inverserh = azimuth('rh',60,150,-15,0)
inverserh =
    238.8595
difference = inverserh-azrh
difference =
    180
```

大圆则不具备这样的特点，如

```
inversegc = azimuth('gc',60,150,-15,0)
inversegc =
    320.9353
difference = inversegc-azgc
difference =
    301.8962
```

几个主要方向上的方位角如表 38-1 所示。

表 38-1　几个主要方向上的方位角

方　向	方 位 角
北	0° 或 360°
东北	45°
东	90°
东南	135°
南	180°
西南	225°
西	270°
西北	315°

仰角是某点所在的局部水平面与该点和另一点连线的夹角。要计算第 1 点到第 2 点的仰角，需要知道两个点的位置和高程。纬度和经度的默认单位为度，高程的默认单位为米。下面计算位置为 10km 东，高程为 10km 的点的仰角

```
[elevang,slantrange] = elevation(0,0,0, 0,km2deg(10),10000)
elevang =
    44.901
slantrange =
    14156
```

结果与平面几何的计算结果有一点点区别，因为地球表面是曲面。

38.5　历年的行星数据

利用工具箱提供的 almanac 函数，可以获得太阳系中主要星体历年的测量数据。这些数据包括太阳、地球和月亮等所有星体的基本几何参数，如椭球体向量、半径、表面积和体积等。

可以获得很多星体的椭球体向量，有些星体只返回球形椭球体向量。

```
almanac('earth','ellipsoid','nauticalmiles')
ans =
     3443.92          0.08
almanac('mars','ellipsoid','kilometers')
ans =
     3396.90          0.11
almanac('moon','ellipsoid','statutemiles')
ans =
     1079.97             0
```

指定'radius'参数时，返回一个标量数据，它表示该星体最佳球形模型的半径。注意，对于球形模型，用弧度表示的半径值为 1。

```
almanac('mercury','radius','kilometers')
ans =
     2439
almanac('neptune','radius','radians')
ans =
     1
```

默认时，表面面积和体积根据球形模型进行计算。在大部分情况下，可以用椭球体模型进行代替，对于地球，可以指定任何系统支持的椭球体模型。下面可以查询地球实际的参数值。

```
almanac('mars','surfarea','kilometers','ellipsoid')
ans =
     1.4441e+08
almanac('earth','volume','kilometers','international')
ans =
     1.0833e+12
almanac('earth','volume','kilometers','actual')
ans =
     1.0832e+12
```

38.6　计算球面四边形的面积

在实体几何中，可以精确计算球面四边形的面积。球面四边形是环形与弓形相交形成的表面图形。用地理术语进行定义，球面四边形是纬线与经线相交形成的区域。

假设有一个球面四边形是 15°N 和 45°N 的纬线与 0° 和 30°E 的经线相交形成的，现在计算它的面积：

```
area = areaquad(15,0,45,30)
area =
    0.0187
```

即，四边形的面积小于星球表面面积的 2%。要获得面积的绝对数值，必须提供球体的半径。地球的半径是 3958.9m，进行计算：

```
area = areaquad(15,0,45,30,3958.9)
area =
3.6788e+06
```

该球面四边形的面积超过了 3 600 000m^2。

第39章 地 图 投 影

所有地理空间数据必须展平到一个表面上，以描绘各种对象的位置。地图投影的数据和图形处理就是为了完成这个任务。尽管对于地理数据投影的方法没有限制，但是规范、限制、标准和应用常常确定了如何使用它。本章介绍什么是地图投影，如何创建它们并进行控制，它们的主要特性以及某些可能性和局限性。

很早以前人们就知道地球是个球体，而不是平面。如果地球真的是平的，则绘图要容易得多，因为地图投影就不需要了。

为了表示地球这样的二维曲面，必须在几何上将这个曲面变换为平面。这个变换就是地图投影。因为许多地图投影不再依赖于物理投影，所以用几何术语来考虑地图投影比较有帮助。因为地图投影与组成圆柱、圆锥和圆这样一些几何对象的点有关，而这些几何对象与要绘图的星球表面上的均质点相对应。

39.1 地图投影的定量属性

球体与多面体、锥体或圆柱不同，它不能展成平面。为了用二维平面描述球面，必须首先定义一个可展表面（即一个可以不通过拉伸或压缩就可以进行分割和展平的表面），并提出将球面的全部或部分对称表示到平面上的规则。任何此类处理都不可避免地要导致这样或那样的变形。地图投影时发生变化主要有5个方面：形状、距离、方向、比例和面积。没有一种投影方法应用于地球表面时能保持一个以上的因素不发生变化。出现这种情况，并不是因为没有提出足够灵活的投影方法，而是因为它在物理上根本不可能实现。上面5个因素可作下面的描述。

1．形状（或者称为保角性）

当地图上任何点在任何方向上的比例尺相同时，形状保持局部不变。具备这种特点的投影称为是保角的。

2．距离（或者称为等距性）

从投影中心到地图上其他任何地方的距离保持不变的地图投影称为是等距的。或者，沿子午线的距离保持不变时也被认为是等距的。

3．方向

在任何方向上方位角都能正确描述时，地图投影保持方向不变。许多方位投影都具有这个特点。

4．比例

比例是图上两点之间的距离与地球表面对应两点之间实际距离的比率。没有一种投影方法能够在很大的范围内保持一种比例不变，但是有些方法可以将这种变化维持在1%～2%。

5. 面积

地图可以按比例描绘地球表面的区域。这样的地图投影方法称为等面积投影。没有一种地图既是等面积又是等角的。

39.2 几何表面

有 3 种标准类型的几何表面可用于地图投影：柱面、锥面和表面。但是，有少数投影类型不能严格归入这 3 类，它们是这 3 种类型的组合。

39.2.1 柱面投影

柱面投影通过覆盖包围象征地球的球体的圆柱来生成，它将地球图像首先投影到柱面上，然后从柱面展开到平面。纬线显示为水平线，经线显示为垂线。图 39-1 显示了柱面投影的原理，左图中地球球体沿赤道与球面相切。

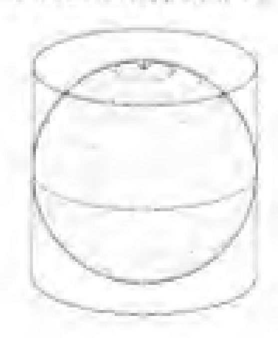



图 39-1 柱面投影

39.2.2 锥面投影

锥面投影是通过将地球表面投影到覆盖它的某个锥面上得到的。锥面的顶点位于地球极轴上，表面与某根纬线相切。多圆锥投影将每个纬带作为单个锥面的一部分，这个锥面沿对应的纬线与地球相切。锥面投影的原理如图 39-2 所示。

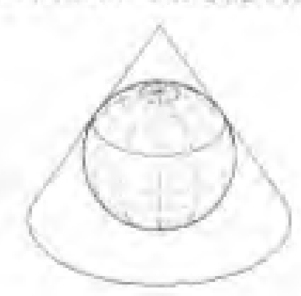



图 39-2 锥面投影

39.2.3 方位投影

方位投影将地球表面投影到平面上。对于极方位投影，平面与地球的某个极点相切，经线投影成放射状的直线段，纬线投影成圆心在极点的同心圆。方位投影的原理如图 39-3 所示。

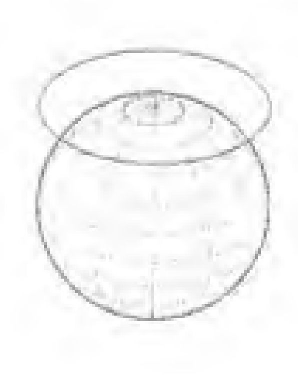
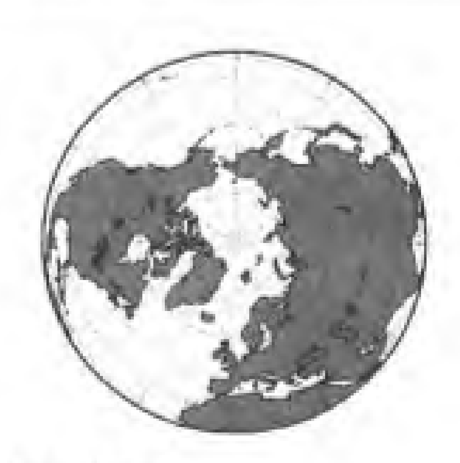



图 39-3　方位投影

上面的每种投影类型都可以作更多的改进：

• 除了与某根标准纬线相切外，柱面和锥面还可以切割两根纬线。同样，用于方位投影的平面也可以与地球相交而不仅仅是相切。

• 投影到几何表面上的中心点可以改变。可以选择从地心或沿极轴、从地球表面的对面或从空间中无限远点进行透视。

39.3　投影方位

到目前为止，对于地图显示的讨论还集中在纵向上，但这不是最常用的投影方位，下面介绍横向、倾斜和歪斜等几个方位的投影。

投影方位主要是针对地图显示而言的，但是，后面也会讨论如何将投影方位作为坐标系信息应用于地图变量，进行地图分析。

39.3.1　origin 向量

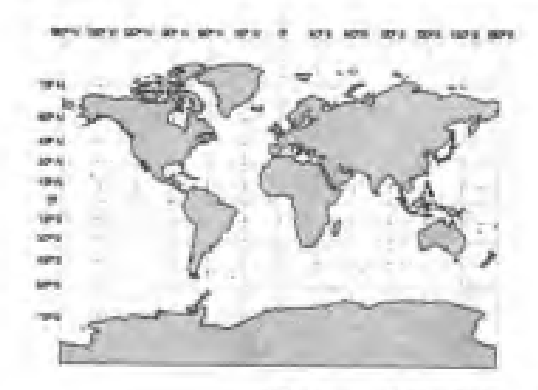
地图坐标系的 origin 属性是一个描述投影几何特征的向量，它具有以下形式：

origvec = [latitude longitude orientation]

其中，latitude 和 longitude 分别表示进行投影计算的中心点的地理坐标。Orientation 为方位角，指向正上方的角度为 0°。该向量的默认值为[0 0 0]，即，投影集中在地理点(0°,0°)上，从这一点出发，北极在正上方的位置。这样的显示是纵向的显示。实际上，只改变 origin 向量的经度值不会改变方位，在纵向方位上，赤道的纬度始终为 0°。

除了具有不同的 origin 向量外，图 39-4 所示的 Miller 投影都是纵向的，其中左图的 origin 向量为[0 0 0]，右图的 origin 向量为[0 -90 0]。

对于纵向而言，柱面沿赤道与地球相切，改变 origin 向量的 longitude 值，对应于绕柱面的中心轴旋转球体。基于对纵向方位投影的理解，可以很容易地理解其他方位的投影。纵向和其他方位的投影如图 39-5 中所示。



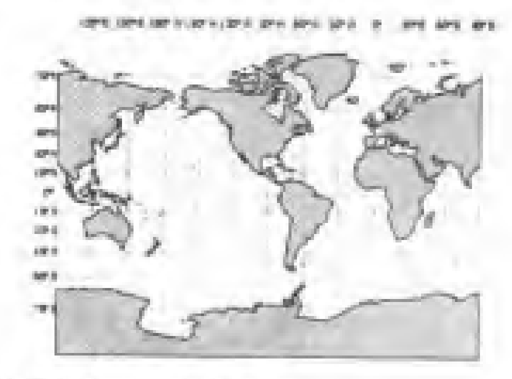


图39-4 对应于不同origin向量的Miller投影结果

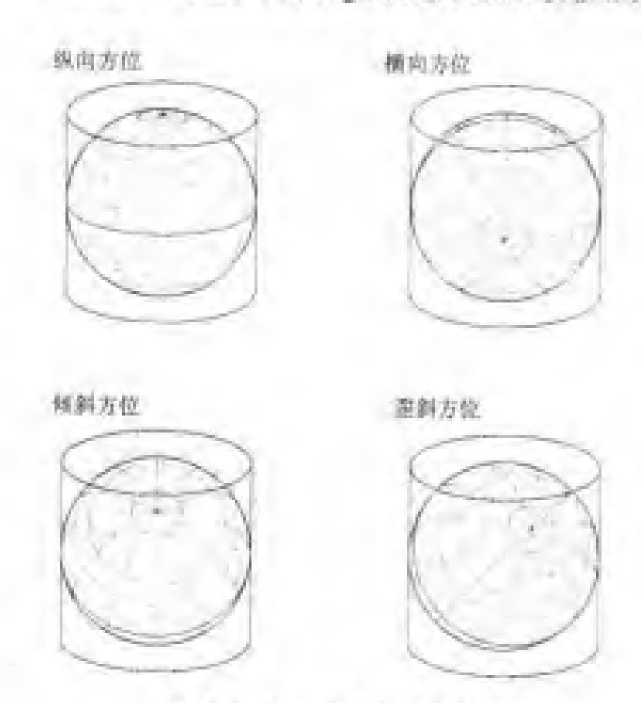


图39-5 纵向和其他方位的投影

当然，几乎没有真正的柱面投影，但覆盖柱面的概念无疑是思考问题的一个便捷方法。为了更好地理解投影，下面介绍几个例子。

下面介绍一个伪柱面投影，使用了正弦曲面。首先查看纵向投影的结果，如图39-6所示。

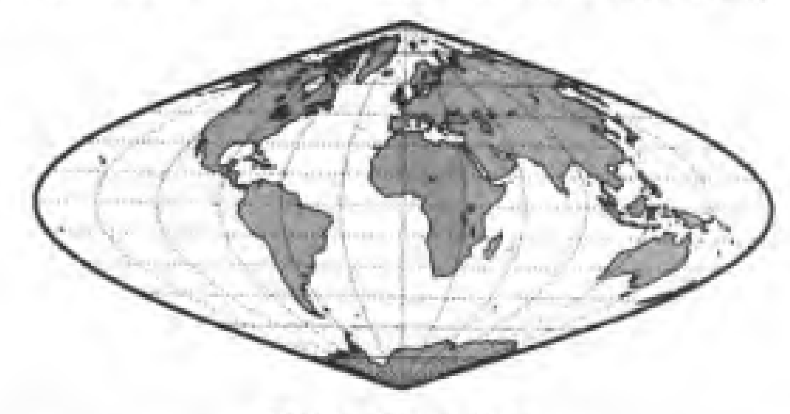


图39-6 纵向投影结果

使用纵向投影时，北极位于图像顶部。为了创建横向方位的投影，假设将北极向下拉，一直拉到图像的中心，它原来的位置为(0°,0°)。图框的形状不变，仍然是正弦曲面投影，如图 39-7 所示。

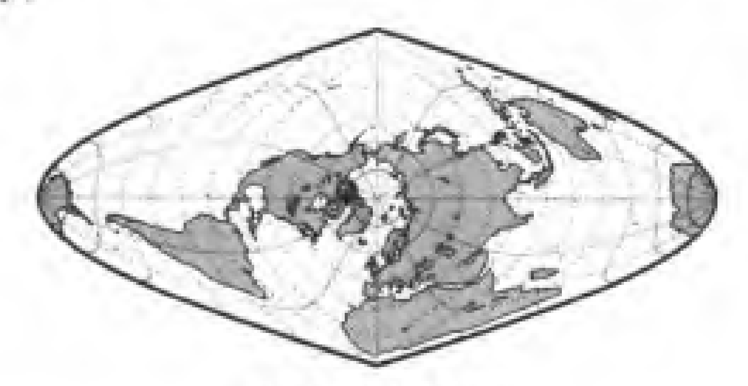


图 39-7　横向投影结果

纵向和横向投影可以认为是极端的情况，界于二者之间的投影就是倾斜投影。在概念上，如果将北极放到纵向投影中的(45°N, 0°)点上，生成的结果将是一个简单的倾斜投影，如图 39-8 所示。

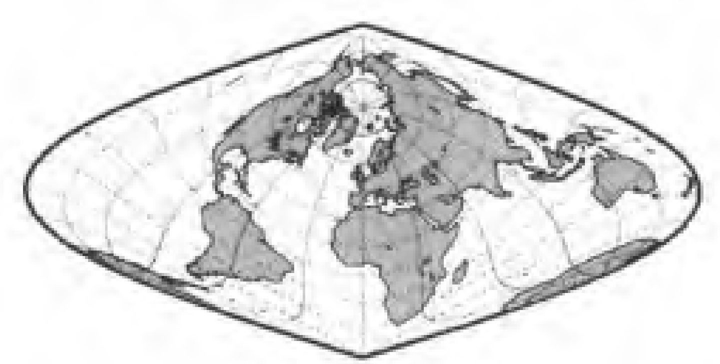


图 39-8　倾斜投影结果

origin 向量为[45 0 45]时的投影结果如图 39-9 所示。为歪斜投影，首先将新原点拉到投影中心(45°N, 0°)，然后旋转，直到北极相对于新原点位于顺时针 45° 的方向上。



图 39-9　歪斜投影结果

不管在哪个方位，投影的特征都要保持下来。就像上面各图所显示的，图像的外观、图框等都不会改变。与方向无关的特征也不会改变，如正弦曲面投影是等面积的，而且在任

何方位上都是这样。与方向有关的特征必须小心考虑，比如，对于纵向的正弦曲面投影，沿任何经线和中心纬线，比例都是不变的，但对于歪斜的情况就不是这样了，歪斜时，沿这些经线和纬线对应的基准投影的线条才是不变的。

基准投影可以认为是标准坐标系，以及与它相一致的纵向投影。其他方位可以认为是坐标变换。

将地图投影导致的变形进行图形显示的一种标准方法是沿球面在均匀间隔的位置上显示小圆。投影以后，小圆显示为不同大小、长度和方向的椭圆，椭圆的大小和形状反映了投影的变形情况。正射投影对应的椭圆是圆，等面积投影对应的椭圆具有相同的面积。可以用命令 tissot 显示这些椭圆，用 clmo Tissot 命令删除它们。如图 39-10 所示。

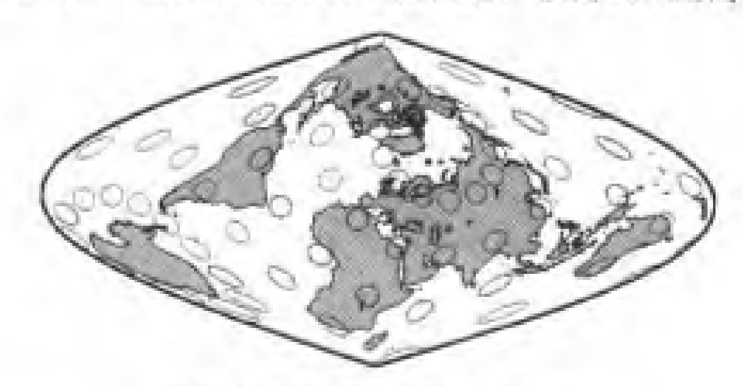


图 39-10 用小圆表示投影导致的变形

使用 mdistort 函数进行显示，是一个更为量化的方法。这个函数在地图上绘制比例、面积或角度变形的等值线。在选择投影类型和投影参数时这种方法很有用。图 39-11 显示了歪斜正弦曲面投影地图中比例变形百分比的等值线，沿基准投影坐标轴变形为 0 的特点在图中表现得很明显。

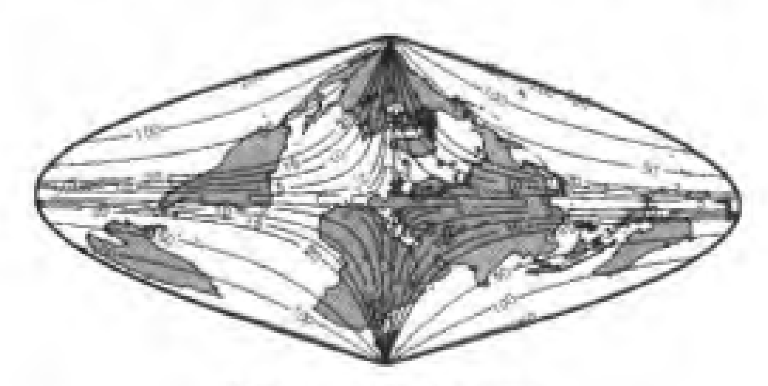


图 39-11 用等值线表示变形

39.3.2 坐标转换

前面讨论了地图投影显示方位的概念，另一个方法是通过重新定义坐标系，然后基于这个新的坐标系显示投影。下面结合实例介绍坐标系的转换。

1. 向量数据--rotatem

假设你现在生活在美国德克萨斯州的米德兰，地理位置为(32°N, 102°W)，你有一个兄弟在塔尔萨(36.2°N, 96°W)，一个妹妹在新奥尔良(30°N, 90°W)，即

```
midl_lat = 32;    midl_lon = -102;
tuls_lat = 36.2; tuls_lon = -96;
newo_lat = 30;    newo_lon = -90;
```

计算他们的家与你家之间的距离如下：

```
dist2tuls = distance(midl_lat,midl_lon,tuls_lat,tuls_lon)
dist2tuls =
    6.5032
dist2newo = distance(midl_lat,midl_lon,newo_lat,newo_lon)
dist2newo =
    10.4727
```

距离单位为度，经纬度的度。大圆方位为

```
az2tuls = azimuth(midl_lat,midl_lon,tuls_lat,tuls_lon)
az2tuls =
    48.1386
az2neworl = azimuth(midl_lat,midl_lon,newo_lat,newo_lon)
az2neworl =
    97.8644
```

这些方位的绝对差为 49.7258°，后面将要用到它。

现在假设你站在世界之颠，所以将米德兰作为转换后坐标系的北极。为了实现这一点，首先确定新坐标系的原点。

```
origin = newpole(midl_lat,midl_lon)
origin =
    58      78      0
```

新坐标系的原点为(58°N, 78°E)，现在米德兰的新纬度为 90°。

塔尔萨和新奥尔良的新坐标是多少呢？下面用 rotatem 命令进行计算。

```
[tuls_lat1,tuls_lon1] = rotatem(tuls_lat,tuls_lon,...
                              origin,'forward','degrees')
tuls_lat1 =
    83.4968
tuls_lon1 =
    -48.1386

[newo_lat1,newo_lon1] = rotatem(newo_lat,newo_lon,...
                              origin,'forward','degrees')
newo_lat1 =
    79.5273
newo_lon1 =
    -97.8644
```

现在用距离和大圆绝对差检验新坐标系。前面我们计算了，米德兰到塔尔萨的距离是 6.5032°，到新奥尔良的距离是 10.4727°。在新坐标系中计算距离很简单，直接将纬度相减就行了，即 90°-83.4968°=6.5032°，90°-79.5273°=10.4727°。另外，米德兰到两个城市

的大圆绝对差为49.7258°，在新坐标系中计算是：-48.1386°-(-97.8644°)=49.7258°。

2．矩阵数据-neworig

可以用矩阵数据创建矩阵地图。假设要将 topo 数据集中的数据转换到新坐标系中，新坐标系中斯里兰卡(7°N, 80°E)为北极。

```
load topo
origin = newpole(7,80);
[map,lat,lon] = neworig(topo,topolegend,origin);
```

显示新地图：

```
axesm miller
surfm(map,[30 30]); demcmap(topo)
```

如图39-12所示。

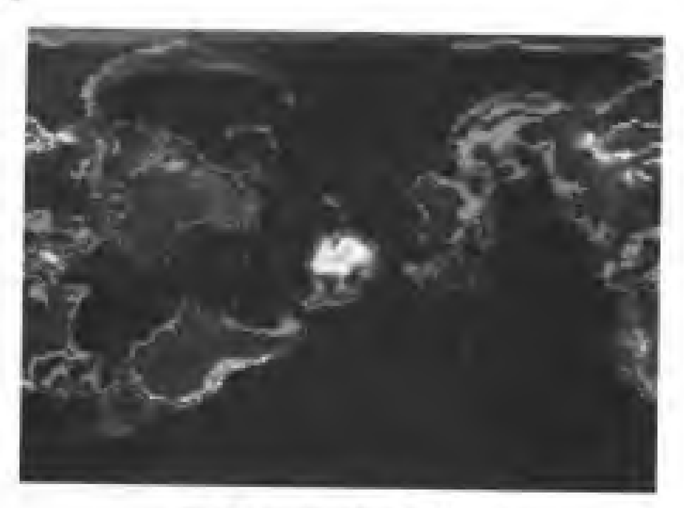


图39-12 新坐标系中的地图

39.4 投影计算

大多数情况下，人们只需要得到地图的图形表示，但有时候又确实需要地图数据在投影空间中的非地理坐标。

用于提取地图投影坐标的简单方法之一就是使用 MATLAB 的 get 命令。投影坐标保存在对象的 XData 和 YData 属性中。作为示例，下面显示 Mollweide 投影的图框，并提取出它的 x-y 坐标。

```
axesm mollweid
h = framem;
x = get(h,'XData');
y = get(h,'YData');
```

当然，不需要通过显示一个地图对象来获取它的投影坐标。可以用工具箱中的显示命令进行相同的投影计算。

下面从显示 coast 向量数据的地图开始。

```
figure
```

```
load coast
plot(long,lat); axis equal
set(gca,'XLim',[-200 200],'YLim',[-100 100])
```

生成图 39-13。注意有些数据超出了 180° 的纬度。

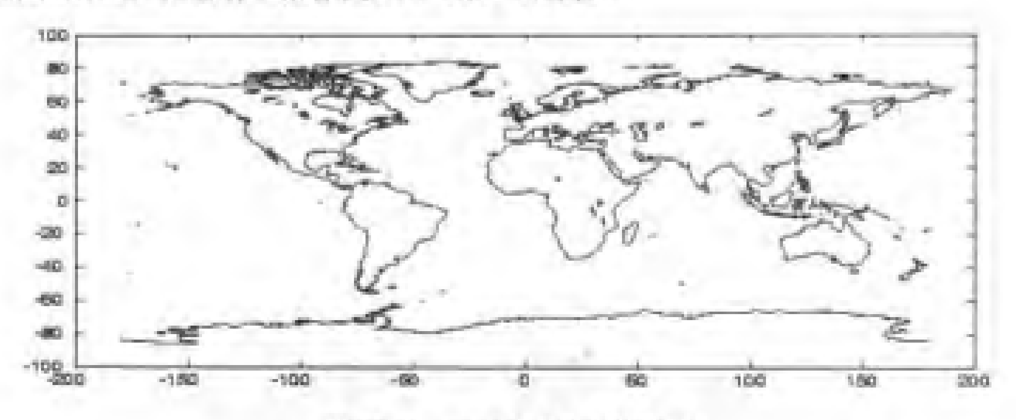


图 39-13　coast 向量数据的地图

投影数据以前，必须定义投影参数。投影参数保存在一个地图投影结构中，该结构通常位于 MATLAB axes 对象的 UserData 属性中，但是可以直接用它进行投影计算。

下面的命令创建一个空的地图投影结构，改变地图原点，用结构的其他字段取默认值。

```
mstruct = defaultm('sinusoid');
mstruct.origin = [0 180 0];
mstruct = defaultm(sinusoid(mstruct));
```

定义地图投影参数以后，可以将经度向量和纬度向量投影到合适的坐标系中并用非映射的方法，即 MATLAB 命令获得结果。

```
[x,y] = mfwdtran(mstruct,lat,long,[],'line');
plot(x,y); axis equal
set(gca,'XLim',[-3.5 3.5],'YLim',[-2 2])
```

生成图 39-14。

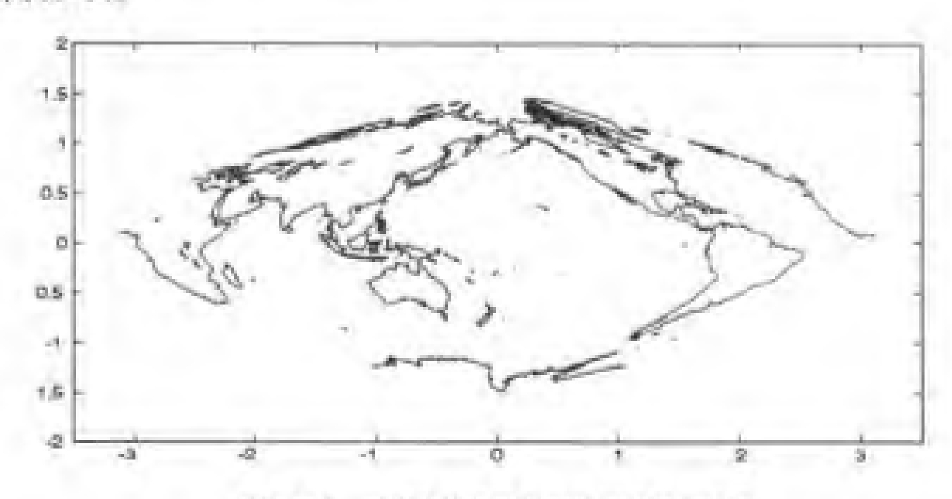


图 39-14　重新定义投影参数以后的显示结果

在转换函数中，将纬度数据对应的参数设置为空矩阵，用'line'表示对象类型，并允许对向量数据进行裁剪。

可以用逆转换函数将投影后的 x-y 数据转换回格林威治地理坐标。

```
[lat2,long2] = minvtran(mstruct,x,y);
plot(long2,lat2); axis equal
set(gca,'XLim',[-200 200],'YLim',[-100 100])
```

生成图 39-15。

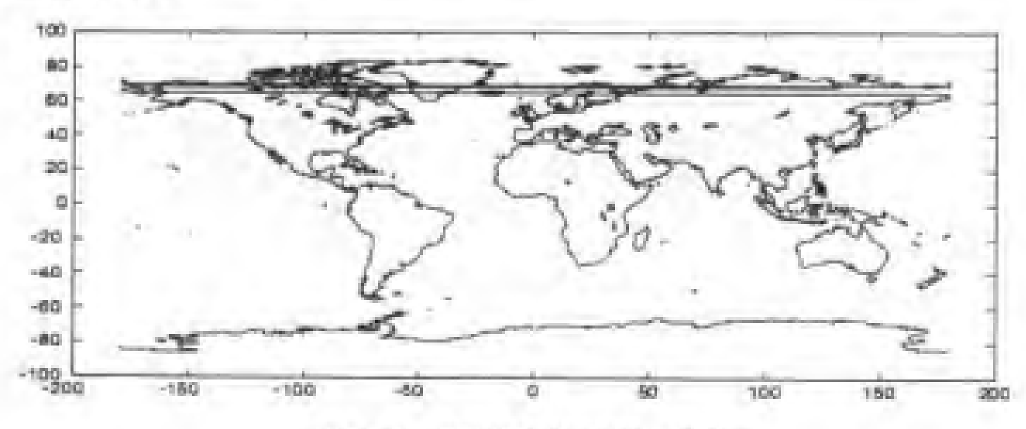


图 39-15 将地图转换回格林威治坐标

现在图 39-13 中超出 180° 范围的数据被调整到[-180 180]中间来了。

除了可以将地理位置投影到笛卡儿坐标外，还可以投影球面与平面之间的角度。对于柱面投影，北面为 y 轴正向，东面为 x 轴正向。在其他投影类型中，这不一定正确。例如，锥面投影中，根据地理坐标的不同，北面可以是 y 轴的左面或右面。

地理上的角度是从北开始，按顺时针方向计算的，而投影角度是从 x 轴开始按逆时针方向计算的。

39.5 使用球面投影

投影是从球坐标系向平面笛卡儿坐标系转换坐标的过程。工具箱中大部分投影的基本假设是纬线和经线向 x 坐标和 y 坐标映射。惟一例外的是球面投影，它将纬线和经线映射为地心笛卡儿三维坐标中球面上的点。这在需要查找对象之间三维关系时很有用。下面的例子演示了二维正交投影与三维球面投影之间的区别。二维正交投影看起来像球面投影，但实际上它是平面的。

```
load topo
axesm ortho; framem
meshm(topo,topolegend);demcmap(topo)
view(3); daspectm('m',1)
[latgrat,longrat] = meshgrat(topo,topolegend,[20 20]);
stem3m(latgrat,longrat,500000*ones(size(latgrat)),'r')
figure
axesm('globe','Geoid',almanac('earth','radius','m'))
```

```
meshm(topo,topolegend); demcmap(topo)
view(3)
[latgrat,longrat] = meshgrat(topo,topolegend,[20 20]);
stem3m(latgrat,longrat,500000*ones(size(latgrat)),'r')
light
```

生成图 39-16，其中左图为二维正交投影结果，右图为三维球面投影结果。




图 39-16　二维正交投影与三维球面投影的比较

因为地球是三维的，在向量数据下面有一个不透明的表面，利用它来隐藏地球另一侧的影像。下面的例子演示如何显示表面下方的线并用相机位置函数控制视图。改变 topo 表面的 FaceColor 属性，可以将地表设置为一种颜色。

```
figure; axesm('globe','galt',0)
gridm('GLineStyle','-')

load topo
hs = meshm(topo,topolegend,size(topo));
hl = displaym(worldlo('POline')); set(hl,'color','k')
demcmap(topo); camlight; material(0.6*[ 1 1 1])
[tlat,tlon] = extractm(worldlo('gazette'),'Moscow');
[plat,plon] = extractm(worldlo('gazette'),'Washington');
camtargm(tlat,tlon,0); camposm(plat,plon,3)
camupm(tlat,tlon); camva(20)
hidem(gca)
```

生成图 39-17。

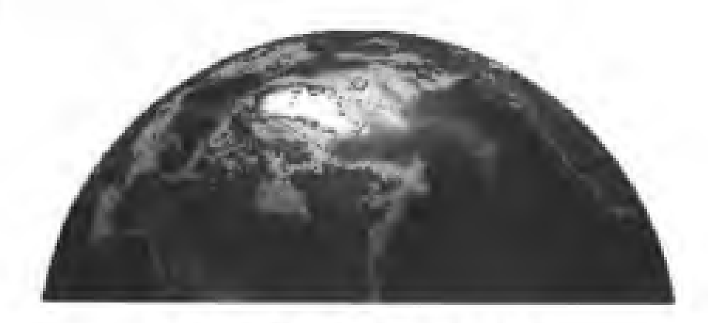


图 39-17　显示数据表面下方的线

39.6 使用UTM投影

通用横向投影（UTM）是科学调查应用中很重要的一种投影方法。该投影将世界分割成一个规则的四边形网格。每个网格的范围为 8°×6°，称为一个区。每个区使用横向的 Mercator 投影，并用椭球体参数限制变形。UTM 投影定义在南 80°和北 84°之间，超出这个范围，使用 UPS 投影。UPS 投影有两个区，北区和南区，它们也有特殊的投影方法和椭球体参数。

使用UTM投影最简单的方法是使用图形用户界面。首先创建地图坐标系。

```
axesm utm
```

生成图 39-18。

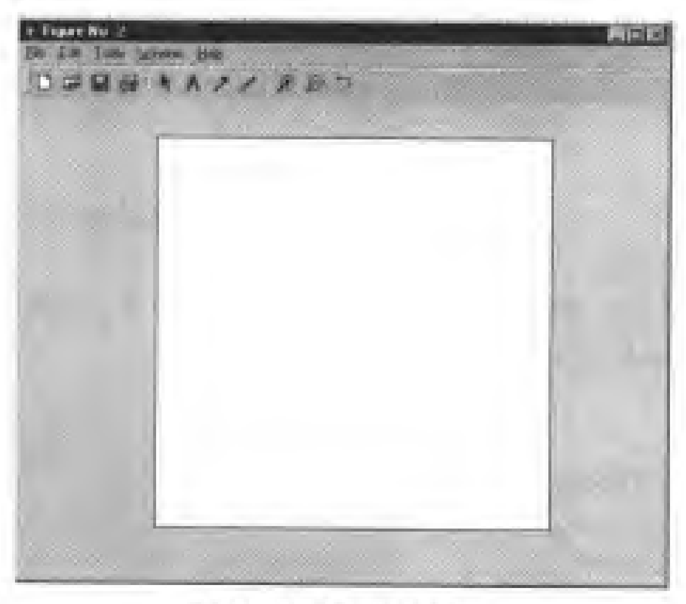


图 39-18 创建地图坐标系

然后使用投影控制面板选择目标区。在命令窗口键入 axesmui 命令可以显示投影控制面板，如图 39-19 所示。在面板上单击“Zone”按钮，打开选择目标区面板，如图 39-20 所示。在目标区附近点击，选择目标区。

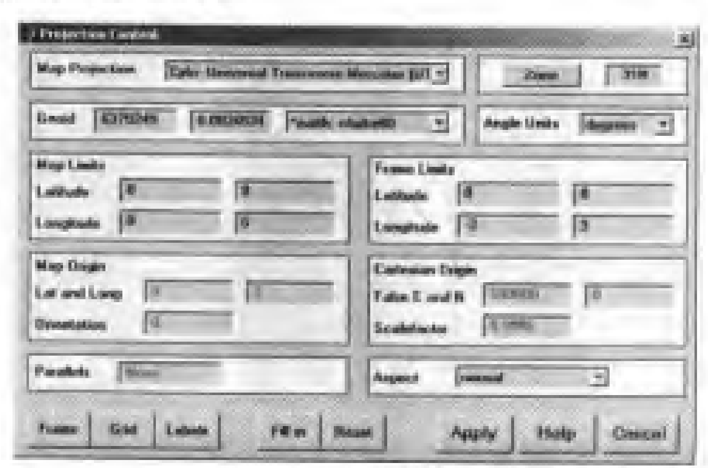


图 39-19 投影控制面板

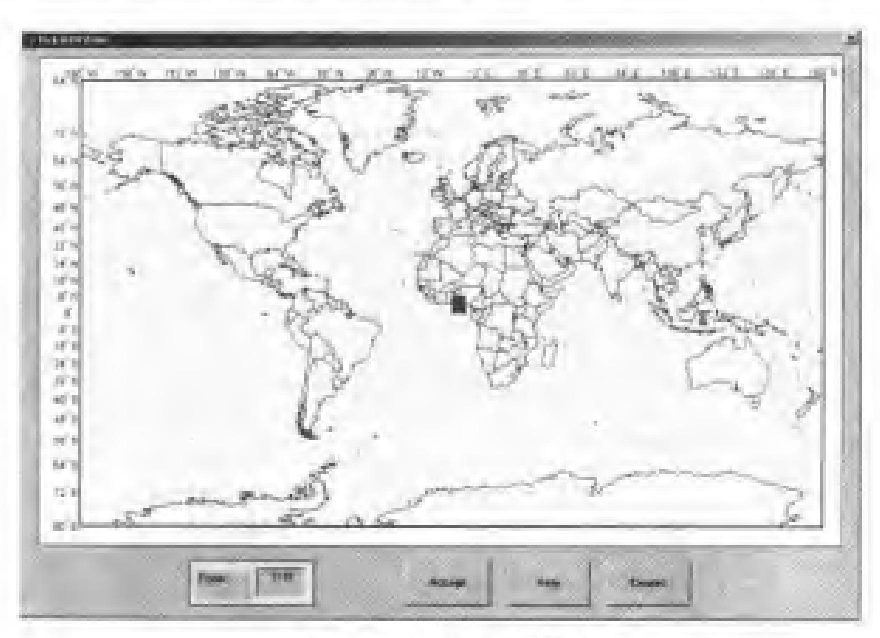


图 39-20　选择目标区面板

单击“Accept”按钮，关闭选择目标区面板。在投影控制面板中进行设置，可以覆盖默认选项。关闭投影控制面板。

现在可以进行投影计算或地图显示了。下面显示 worldlo 和 worldhi 地图集中的数据。

```
displaym(worldhi([40 48],[-78 -72]))
polcmap
framem
displaym(worldlo('PPpoint'))
displaym(worldlo('PPtext'))
trimcart alltext; tightmap
hidem(gca)l set(gcf,'color','w')
```

生成图 39-21。

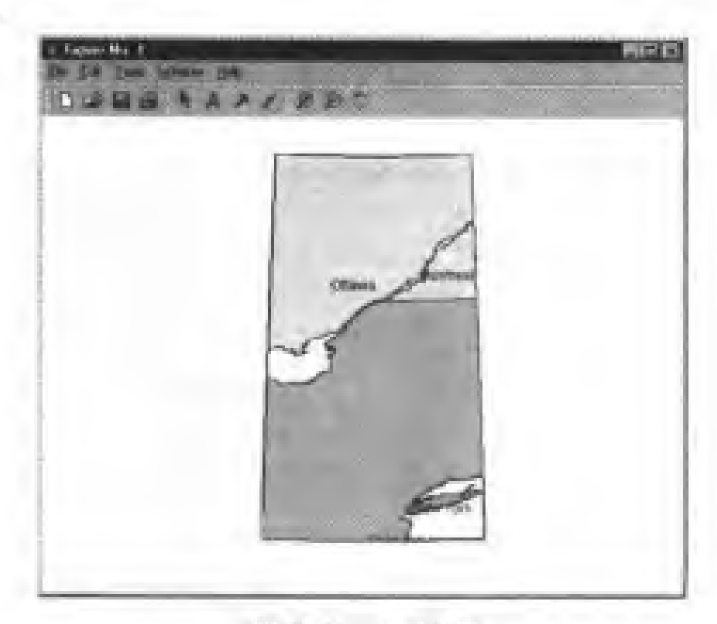


图 39-21　显示地图

还可以根据纬度和经度计算投影后的UTM网格坐标。

```
[x,y] = mfwdtran(40.5,-73.5)
x =
      627106.47
y =
     4484124.43
```

当然，可以在不显示地图的情况下使用UTM投影计算坐标。例如，

```
utmzone(40.5,-73.5)
ans =
18T
[ellipsoid,estr] = utmgeoid('18T')
ellipsoid =
    6378206.40            0.08
estr =
clarke66

close all
mstruct = defaultm('utm');
mstruct.zone = '18T';
mstruct.geoid = almanac('earth','geoid','m','clarke66');
mstruct = defaultm(utm(mstruct));
[x,y] = mfwdtran(mstruct,40.5,-73.5)
x =
      627106.47
y =
     4484124.43
```

下面的例子同时显示多个UTM区。

```
latlim = [-60 -15];centralMeridian = -70; widdı = 20;
axesm('mercator',...
     'Origin',[0 centralMeridian -90],...
     'Flatlimit',[-width/2 width/2],...
     'Flonlimit',sort(-latlim),...
     'Aspect','transverse')
displaym(worldlo('POline'));framem
gridm; setm(gca,'plinefill',1000)
tightmap
mdistort scale
```

生成图39-22。

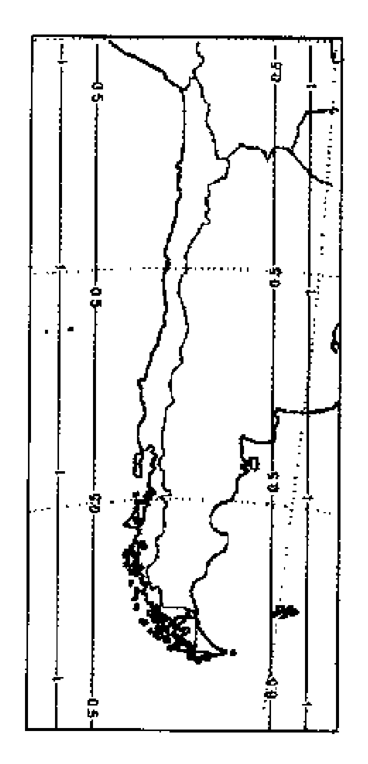


图39-22 显示多个UTM区

39.7　投影类型综述

从严格意义上讲，可以创建无数种可能的地图投影。地图制作工具箱提供了 60 种不同的地图投影方法。在命令窗口输入 maps 命令，可以查看所有的投影类型。

现将这 60 种投影类型及其性质归纳在表 39-1 中，具体内容可以参见对应函数的帮助文档。

表 39-1　投影类型及其性质

投影类型	语法	类型	等面积	等角	等距	特殊性质
Balthasart	balthsrt	柱面投影	●			
Behrmann	behrmann	柱面投影	●			
Bolshoi Sovietskii Atlas Mira	bsam	柱面投影				
Braun Perspective	braun	柱面投影				
Cassini	cassini	柱面投影			●	
Central	ccylin	柱面投影				
Equal-Area Cylindrical	eqacylin	柱面投影	●			
Equdistant Cylindrical	eqdcylin	柱面投影			●	
Gall Isographic	giso	柱面投影			●	
Gall Orthographic	gortho	柱面投影	●			
Gall Stereographic	gstereo	柱面投影				
Lambert Equal-Area Cylindrical	lambcyln	柱面投影	●			
Mercator	mercator	柱面投影		●		1
Miller	miller	柱面投影				
Plate Carree	pcarree	柱面投影			●	
Trystan Edwards	trystan	柱面投影	●			
Universal Transverse Mercator (UTM)	utm	柱面投影		●		
Wetch	wetch	柱面投影				
Apianus II	apianus	伪柱面投影				
Collignon	collig	伪柱面投影	●			
Craster Parabolic	craster	伪柱面投影	●			
Eckert I	eckert1	伪柱面投影				
Eckert II	eckert2	伪柱面投影	●			
Eckert III	eckert3	伪柱面投影				
Eckert IV	eckert4	伪柱面投影	●			
Eckert V	eckert5	伪柱面投影				
Eckert VI	eckert6	伪柱面投影	●			
Fournier	fournier	伪柱面投影	●			
Goode Homolosine	goode	伪柱面投影	●			
Hatano Assymetrical Equal-Area	hatano	伪柱面投影	●			
Kavraisky V	kavrsky5	伪柱面投影	●			
Kavraisky VI	kavrsky6	伪柱面投影	●			
Loximuthal	loximuth	伪柱面投影				2
McBryde-Thomas Flat-Polar Parabolic	flatplrp	伪柱面投影	●			

续表

投影类型	语法	类型	等面积	等角	等距	特殊性质
McBryde-Thomas Flat-Polar Quartic	flatplrq	伪柱面投影	●			
McBryde-Thomas Flat-Polar Sinusoidal	flatplrs	伪柱面投影	●			
Mollweide	mollweid	伪柱面投影	●			
Putnins P5	putnins5	伪柱面投影				
Quartic Authalic	quartic	伪柱面投影	●			
Robinson	robinson	伪柱面投影				
Sinusoidal	sinusoid	伪柱面投影	●			
Tissot Modified Sinusoidal	modsine	伪柱面投影	●			
Wagner IV	wagner4	伪柱面投影	●			
Winkel I	winkel	伪柱面投影				
Albers Equal-Area Conic	eqaconic	锥面投影	●			
Equidistant Conic	eqdconic	锥面投影			●	
Lambert Conformal Conic	lambert	锥面投影		●		
Murdoch I Conic	murdoch1	锥面投影			●	3
Murdoch III Minimum Error Conic	murdoch3	锥面投影			●	3
Bonne	bonne	伪锥面投影	●			
Werner	werner	伪锥面投影	●			
Polyconic	polycon	伪锥面投影				
Van Der Grinten I	vgrint1	伪锥面投影				
Breusing Harmonic Mean	breusing	方位投影				
Equidistant Azimuthal	eqdazim	方位投影			●	
Gnomonic	gnomonic	方位投影				4
Lambert Azimuthal Equal-Area	eqaazim	方位投影	●			
Orthographic	ortho	方位投影				
Stereographic	stereo	方位投影		●		5
Universal Polar Stereographic (UPS)	ups	方位投影		●		5
Vertical Perspective Azimuthal	vperspec	方位投影				
Wiechel	wiechel	伪方位投影	●			
Aitoff	aitoff	修正的方位投影				
Briesemeister	bries	修正的方位投影	●			
Hammer	hammer	修正的方位投影	●			
Globe	globe	球面投影	●	●	●	6

表39-1中，特殊性质一栏中的数字说明如下：

1—直恒向线；

2—从中心点出来的恒向线都是直的，比例和方位不变；

3—总面积不变；

4—直线大圆；

5—大圆和小圆看起来像圆或直线；

6—三维显示。

第 40 章　创建和查看地图

地图制作工具箱提供了很多方法来控制地理空间数据的显示。下面介绍一些比较重要的函数和显示向量数据和栅格数据并与它们进行交互的相关界面。

40.1　地图制作简介

使用地图制作工具箱显示地理信息可以像 MATLAB 中绘制表格或时间序列数据图形一样容易。除了接收地理/测量坐标数据以外，大部分制图函数与 MATLAB 绘图函数近似。地图制作函数尤其如此，它们与 MATLAB 中具有相似功能的函数具有相同的名称，只是加上了后缀“m”。例如，地图制作工具箱的 plotm 函数就类似于 MATLAB 的绘图函数 plot。

地图制作工具箱管理当前地图中的大部分细节。它投影数据、将数据裁剪到合适的大小以适合指定的范围，并以不同比例显示最后生成的地图。使用该工具箱，还可以添加一般图形具备的元素，如图框、网格线、坐标标注和文本标注等到当前地图中。如果改变投影属性，或甚至改变投影类型，则地图制作工具箱用新设置重绘地图。

工具箱还使地图的修改和操作更加容易。可以从命令行或图形用户界面和属性编辑工具修改地图的显示特性和地图中的对象。大部分地图显示函数都有图形用户界面。

40.1.1　用 worldmap 和 usamap 函数显示简单的地图

使用地图制作工具箱可以控制地图显示的许多方面。有些函数接受多个参数，但是几乎所有参数都有默认值。有些函数，例如 worldmap 和 usamap，控制很容易。通过自动选择合适的地图投影类型和设置，这两个函数可以创建世界地区地图和美国地图。然后可以用后面介绍的方法修改或添加地图显示内容。也可以用函数进行地图显示和操作。

这里有两个创建地图的例子。第 1 个创建南非的轮廓图。

```
figure
worldmap 'south africa'
hidem(gca)
```

图形效果如图 40-1 所示。

第 2 个例子通过指定地理范围，创建切萨皮克湾地区的地图，另外还设置了文本标注。

```
latlim = [37 40]; lonlim = [-78 -74];
figure
usamap(latlim,lonlim)
textm(38.2,-76.1,' Chesapeake Bay ',...
'fontweight','bold','Rotation', 270)
```

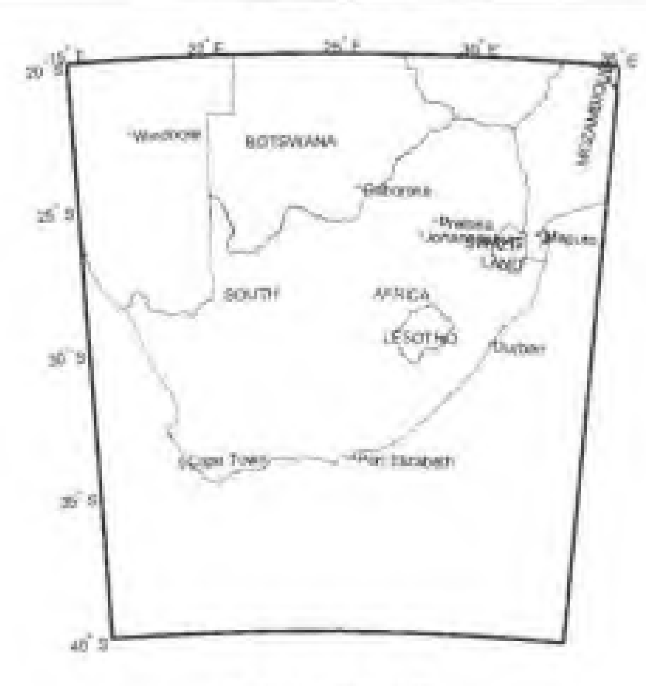


图 40-1　南非地图

生成图 40-2。

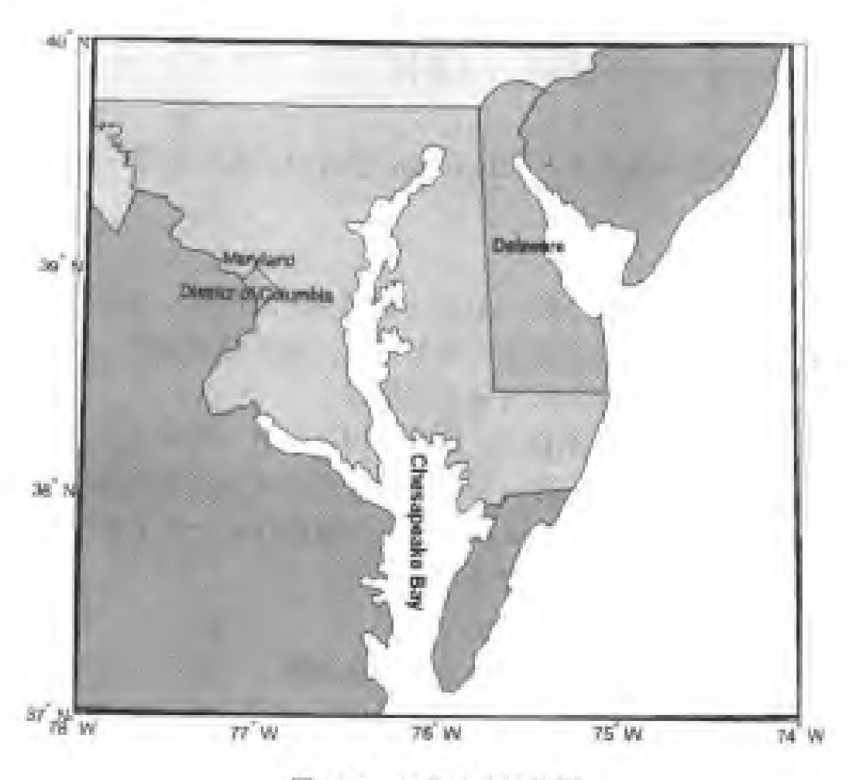


图 40-2　切萨皮克湾地图

40.1.2　坐标

可以用地图制作工具箱中的任何一个内部界面，或者 MATLAB 和地图制作工具箱函数生成地图。许多 MATLAB 图形都是用 axes 函数生成的，例如：

```
axes
axes('PropertyName',PropertyValue,...)
axes(h)
h = axes(...)
```

地图制作工具箱提供了 axes 函数的一个扩展版本，称为 axesm，它包括了当前坐标系统的信息。axesm 函数的语法为

```
axesm
axesm(handle)
axesm(PropertyName,PropertyValue,...)
axesm(ProjectionFile,PropertyName,PropertyValue,...)
```

调用没有变量的 axes 函数会弹出一个用户界面，它列出了所有支持的投影类型并帮助定义它们的参数。可以用 axesmui 函数激活这个图形界面。

还可以用 maps 函数列出地图制作工具箱地图投影的名称、类型和 ID 字符串。axesm 函数创建的坐标系与 MATLAB 中创建的坐标系具有相同的属性，另外还有与投影、比例和地理坐标位置有关的属性。

axesm 函数创建的地图坐标系将投影信息包含在一个可以通过它们的 UserData 属性获取的结构中。例如，键入下面的命令行可以查看所有相关属性。

```
h = axesm('MapProjection','mercator')
```

然后用 getm 函数提取出所有地图坐标系属性。

```
p = getm(h)
```

因为投影数据保存在坐标系结构的 UserData 字段中，也可以用一般的坐标系属性获得它。

```
q = get(h, 'UserData')
```

用 axesm 函数创建的图形窗口包括与任何 MATLAB 图形窗口相同的工具和菜单集，默认时是空的，即使工作空间中有地图数据时也是如此。对于某些属性，如网格、图框和坐标系标注等可以通过右键编辑功能进行打开和关闭操作。

可以定义多个独立的地图坐标系图形，但任何时候只有一个是激活的。

就像使用 MATLAB 函数 set 和 get 可以获取和操作标准坐标系的属性一样，地图坐标系属性也可以用函数 setm 和 getm 进行获取和操作。下面结合一个例子进行介绍。

（1）创建一个不包含地图数据的地图坐标系。

```
axesm('MapProjection','miller','Frame','on')
```

（2）键入下面的命令行，获取当前的 FlineWidth 属性。

```
getm(gca,'FLineWidth')
ans =
    2
```

（3）现在将直线的宽度重新设置为 4 磅。默认的度量单位是磅，可以设置为英寸、厘米或像素等其他单位。

```
setm(gca,'FLineWidth',4)
```

（4）可以用 setm 函数同时设置任意个属性。继续上面的例子，减小线宽，将投影类型

改变为等距柱面投影。

```
setm(gca,'FLineWidth',3,'MapProjection','eqdcylin')

getm(gca,'FLineWidth')
ans =
    3
getm(gca,'MapProjection')
ans =
eqdcylin
```

（5）用下面的命令探察当前设置下的所有地图坐标系数属性。

```
getm(gca)
ans =
     mapprojection: 'eqdcylin'
              zone: []
        angleunits: 'degrees'
            aspect: 'normal'
      falseeasting: []
     falsenorthing: []
       fixedorient: []
   ⋮
       plabelround: 0
```

（6）类似地，可以用 setm 函数显示属性集，包括它们的枚举值和默认值等。

```
setm(gca)
AngleUnits              [ {degrees} | radians | dms | dm ]
Aspect                  [ {normal} | transverse ]
FalseEasting
FalseNorthing
⋮
PLabelMeridian
PLabelRound
```

40.1.3 在投影类型之间转换

一旦用 axesm 函数创建了坐标系，不管地图数据是否显示，都可以改变当前投影类型及其参数。可以用 setm 或 maptool 图形界面重新定义投影。如果这样做，可能需要改变有些坐标属性以获得合适的外观。

尽管相似的投影可以有同样的属性集，但其他一些可能完全不同。地图投影的类型往往可以暗示是否需要进行修改。例如，柱面投影转换到方位投影只需要作很少的修改，下面的例子对此进行演示。

（1）创建有子午线平行标注的 Mercator 投影。

```
axesm mercator
```

```
framem on; gridm on; mlabel on; plabel on
setm(gca,'LabelFormat','signed')
```

生成图 40-3。

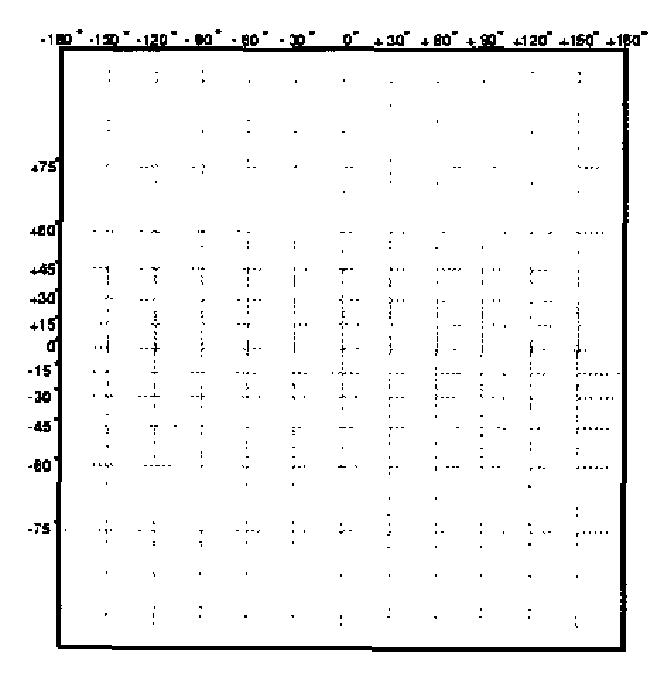


图 40-3　有子午线标注的 Mercator 投影

（2）获取默认的地图和图框的纬度范围限制。

```
[getm(gca,'MapLatLimit'); getm(gca,'FLatLimit')]
ans =
-86      86
-86      86
```

（3）将投影类型转换为正交方位投影。

```
setm(gca,'MapProjection','ortho')
```

（4）对于正交投影，必须将图框和地图范围手工重设为合适的值，以显示圆形图框。如果不知道它们的默认值或什么值合适，可以给任何属性值指定一个空矩阵。

```
setm(gca,'FLatLimit',[],'MapLatLimit',[])
[getm(gca,'MapLatLimit'); getm(gca,'FLatLimit')]
ans =
    -90      90
    -Inf 89
```

（5）还需要手工指定子午线平行标注的位置。

```
setm(gca,'MLabelParallel',0,'PLabelMeridian',-90)
```

现在的地图如图 40-4 所示。

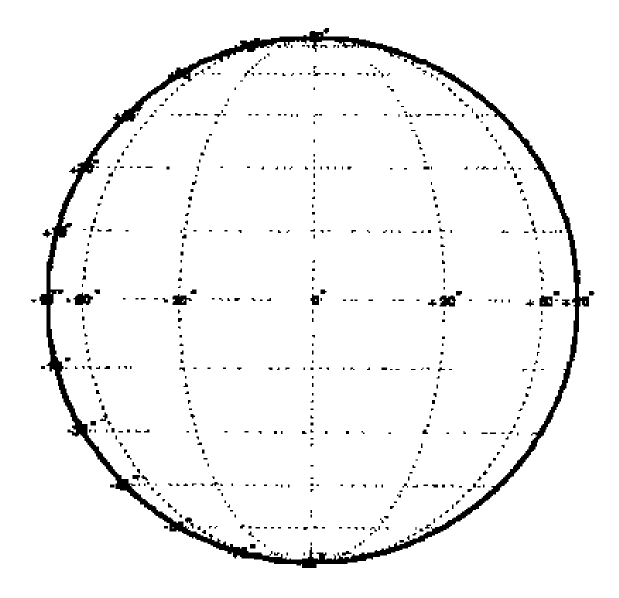


图 40-4　正交投影显示

40.2　用地图制作工具箱函数显示向量数据

除了用 mapview、maptool 和其他地图制作工具箱图形用户界面创建地图以外，还可以通过输入命令或使用脚本交互创建地图。本节介绍如何用主要的地图制作函数显示向量地理空间数据。下面介绍显示栅格地图数据的方法。

40.2.1　把向量地图显示成直线对象

使用地图制作工具箱可以把向量地图数据显示成直线对象。相关函数如表 40-1 所示。

表 40-1　把向量地图数据显示成直线对象的相关函数

函　数	功　能
contourm	绘地图数据的等值线图
contour3m	绘 3 维空间中地图数据的等值线图
linem	绘投影到地图坐标系中的直线对象
plotm	清除图形窗口中的内容并绘投影到地图坐标系的直线对象
plot3m	在 3 维空间中将直线对象投影到地图坐标系

下面结合一个例子进行介绍。

（1）设置地图坐标系和图框。

```
load coast
axesm mollweid
framem('FEdgeColor','blue','FLineWidth',0.5)
```

（2）用 plotm 函数绘制 coast 向量数据的图形，可以在命令窗口指定直线对象的属性名和值。

```
plotm(lat,long,'LineWidth',1,'Color','blue')
```

有时候向量数据表示特定点。假设用变量表示开罗、里约热内卢和佩斯的位置，并且现在只想用标记表示它们，不用直线段进行连接。

（3）定义 3 个城市的地理位置并用标记标出它们。

```
citylats = [30 -23 -32]; citylongs = [32 -43 116];
plotm(citylats,citylongs,'r*')
```

图形效果如图 40-5 所示。

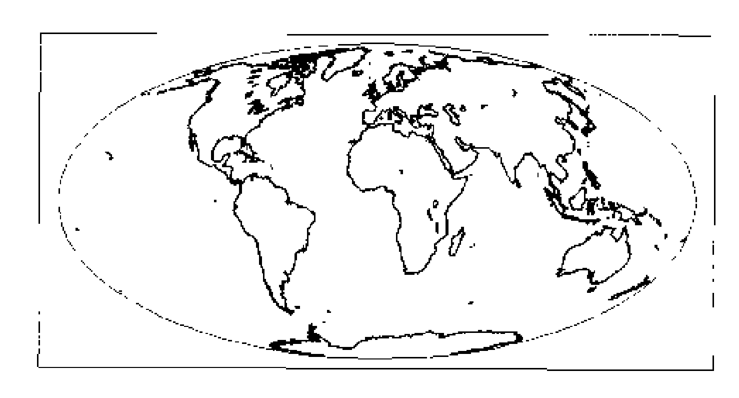


图 40-5　标出 3 个城市的位置

（4）除了这些“永久”的地理数据外，还可以显示计算得到的向量数据。计算并显示从开罗到里约热内卢的大圆和开罗到佩斯的恒向线。

```
[gclat,gclong] = track2('gc',citylats(1),citylongs(1),...
                        citylats(2),citylongs(2));
[rhlat,rhlong] = track2('rh',citylats(1),citylongs(1),...
                        citylats(3),citylongs(3));
plotm(gclat,gclong,'m-'); plotm(rhlat,rhlong,'m-')
```

效果如图 40-6 所示。

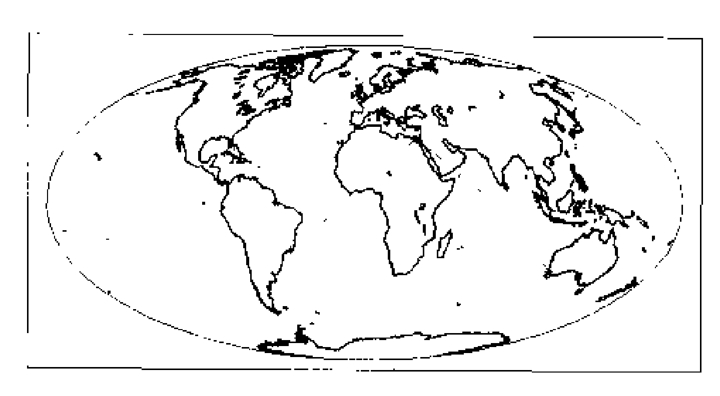


图 40-6　显示大圆和恒向线

40.2.2　把向量地图显示成面片

向量地图数据可以显示为面片或填充多边形。patchm 函数是地图制作工具箱中与 MATLAB 中的 patch 函数等价的函数。

表 40-2 列出了地图制作工具箱中可用的面片显示函数。

表 40-2　面片显示函数

函　数	功　能
fillm	填充二维地图多边形
fill3m	填充三维空间中的三维地图多边形
patchm	投影到地图坐标系的面片对象
patchesm	作为单独对象投影到地图坐标系的面片

（1）首先载入 usalo 数据集，用 who 命令查看它的结构。

```
load usalo
who
Your variables are:

conus          gtlakelon     statelat      uslon
greatlakes     state         statelon
gtlakelat      stateborder   uslat
```

变量 uslat 和 uslon 一起描述 3 个多边形，大多边形用来表示美国的轮廓，两个小一些的表示美国的另外两处陆地。变量 gtlakelat 和 gtlakelon 描述 3 个表示美国 5 大湖的多边形。变量 statelat 和 statelon 包含表示州界的直线段。

（2）键入下面的命令行，核实直线和面片数据中包含 NaN。

```
find(isnan(gtlakelon))  %or greatlakelat
ans =
        883
       1058
       1229
```

（3）由此可以知道 greatlakes 变量包括 3 个地理对象。输入下面的命令行，可以知道这 3 个对象是面片。

```
greatlakes
greatlakes =
1x3 struct array with fields:
    type
    tag
    lat
    long
    altitude
    otherproperty
```

可见它是一个有 6 个字段的结构。

（4）Type 字段指定数据保存为直线还是面片。

```
greatlakes.type
ans =
patch
```

（5）提供一个空矩阵，用平行线作为默认的纬线。

```
axesm('MapProjection','eqaconic', 'MapParallels',[],...
        'MapLatLimit',[23 52], 'MapLonLimit',[-130 -62])
```

（6）如果不知道什么纬度和经度范围适合于地图，开始时可以使用最小值和最大值，如下所示。

```
uslatlim = [min(uslat) max(uslat)]
uslatlim =
   25.1200    49.3800

uslonlim = [min(uslon) max(uslon)]
uslonlim =
 -124.7200   -66.9700
```

（7）然后可以用这些变量精确设置图框的范围限制，它将剔除地图周围的所有边界。

```
axesm('MapProjection','eqaconic', 'MapParallels',[],...
        'MapLatLimit',uslatlim, 'MapLonLimit',uslonlim)
```

显示面片数据时，分图层显示就很重要，因为上层的对象会覆盖下层的对象。利用各层的显示次序和高度可以控制对象的可见性。所以，利用地图制作工具箱中的主要面片显示函数 fillm 可以控制所显示的面片的 z 轴水平。这里，陆地数据显示在默认水平上，即 z=0；5 大湖数据指定为 z=1。

（8）指定地平纬度为 0.5 度，用 plot3m 函数绘制各层之间的直线段图。

```
fillm(uslat,uslon, 'FaceColor',[1 .5 .3], 'EdgeColor','none')
fillm(gtlakelat,gtlakelon,1,...
   'FaceColor','cyan', 'EdgeColor','none')
plot3m(statelat,statelon,0.5,'k')
```

图 40-7 是生成的地图。



图 40-7　生成的地图

第41章　制作三维地图

利用地图制作工具箱可以创建三维地图。任何地图都可以创建和显示成三维的，有些主题地图制作函数可以绘制三维标记。最常用的三维应用是地形可视化，图中用地形数据网格表示海拔高度。本章介绍如何获取和操作地形数据，制作三维表面的技巧和在地形上覆盖其他数据的方法，以及如何加阴影、光照和进行平面和球形的三维地貌显示。

41.1　地形数据源

几乎所有已经发布的地形高程数据都是数据网格形式的。下面介绍一些地形数据的通用格式，以及如何为目标区域获取和准备数据集。

41.1.1　源于NIMA的数字地形高程

数字地形高程（DTED）模型是一系列网格化的高程模型，分辨率为 1 km 或更高。DTED 模型是美国国家图片和制图局（NIMA）的产品，NIMA 的前身是国防制图局（DMA）。

1. 0级DTED文件

分辨率最低的数据是 0 级 DTED，网格间隔为 1 公里左右。NIMA 将 DTED 规范发布在 http://www.nima.mil/ast/fm/acq/89020.pdf 上。通过互联网，可以从 NIMA 下载 0 级 DTED 文件。在网址 http://earth-info.nima.mil/上可以查看 NIMA 的公共数据浏览器。DTED 文件是二进制文件，文件名的扩展名为 dtN，其中，N 表示 DTED 产品的级别。

2. 更高分辨率的DTED文件

NIMA 还提供了更高分辨率的地形数提文件。1 级 DTED 文件的分辨率是 100 米左右，是 USGS 1∶250 000 DEM 模型的主要数据源。2 级和 2 级以上的 DTED 文件具有更高的分辨率。

41.1.2　源于USGS的数字高程模型（DEM）文件

美国地质勘察局（USGS）为美国适合在 1∶24 000 至 1∶250 000 比例尺之间使用的地图准备了地形数据网格。这些数据中有些源于 DTED 模型。

比例尺最大的 USGS DEM 被进行了分割，以便于与 USGS 1∶24 000 比例尺的地图系列相匹配。这些高程模型的网格间距是 30m，每个文件覆盖 7.5 个小四边形网格。

41.1.3　确定区域内存在什么高程数据

地图制作工具箱提供了几个函数和 GUI 来帮助用户获取目标区域 DEM 数据的文件名

并进行管理。表 41-1 中描述了读取数据、确定目标区域可能存在的文件名或返回高程网格文件元数据的函数。

表 41-1　对应于不同文件类型的函数

文件类型	描　述	读取文件的函数	识别文件的函数
DTED	美国国防部的数字地形高程数据	dted	dteds
DEM	USGS 1 度（100m 的分辨率）数字高程模型	usgsdem	usgsdems
DEM24K	USGS 1：24 000 比例尺（30m 的分辨率）数字高程模型	usgs24kdem	n.a.
GTOPO30	1000 米分辨率的全局高程模型	gtopo30	gtopo30s
SDTS DEM	美国 SDTS 格式的数字高程模型	sdtsdemread	sdtsinfo

注意，识别文件名的函数的名称是在各自对应的文件数据读取函数名后面加 s。这些函数确定目标区域的文件名，调用时的变量形式为（latlim, longlim），它确定目标区域纬度和经度的范围限制，并且都返回一个提供所需高程的文件名列表。在 latlim 和 longlim 中，最南面的纬度和最西面的经度必须分别放在第 1 的位置上。

1．用 dteds、usgsdems 和 gtopo30s 函数识别 DEM 文件

首先定义目标区域的范围，即纬度 41.1°N 至 43.9°N，经度 71.9°W 至 69.1°W。

（1）用 dteds 函数确定需要哪个 DTED 文件，该函数返回一个字符串单元数组。

```
dteds([41.1 43.9],[-71.9 -69.1])
ans =
    '\DTED\W072\N41.dt0'
    '\DTED\W071\N41.dt0'
    '\DTED\W070\N41.dt0'
    '\DTED\W072\N42.dt0'
    '\DTED\W071\N42.dt0'
    '\DTED\W070\N42.dt0'
    '\DTED\W072\N43.dt0'
    '\DTED\W071\N43.dt0'
    '\DTED\W070\N43.dt0'
```

（2）用 usgsdems 函数确定需要的 USGS DEM 文件。

```
usgsdems([41.1 43.9],[-71.9 -69.1])
ans =
    'portland-w'
    'portland-e'
    'bath-w'
    'boston-w'
    'boston-e'
    'providence-w'
    'providence-e'
    'chatham-w'
```

（3）用 gtopo30s 函数确定需要的 GTOPO30 文件。

```
gtopo30s([41.1 43.9],[-71.9 -69.1])
ans =
    'w100n90'
```

2. 用 DTED 函数绘制单个 DTED 文件中的地图

下面的例子给一幅海角地图的 0 级 DTED 数据着色。

（1）定义目标区域并确定要获取的文件。

```
mylats = [41.2 41.95]
mylons = [-70.95 -70.1]
dteds(mylats, mylons)
ans =
    'dted\w071\n41.dt0'
```

（2）从 NIMA 或 MATLAB CD-ROM 中下载包含此文件的目录。

（3）用 dted 函数在工作空间中创建一个地形网格。如果路径中有 1 个以上的 DTED 文件名为 n41.dt0，则工作目录必须是/dted/w071，以确保 dted 函数找到正确的文件。如果文件不在路径上，会提示你用 dted 函数得到 n41.dt0 文件，即

```
[cape1,caperef1] = dted('n41.dt0',1,mylats,mylons);
```

（4）因为 DTED 文件不包括海洋深度信息，略微降低 0 高程，以便颜色查找表重置时给它们着上蓝色。

```
cape1(cape1==0)=-1;
```

（5）用 usamap 函数生成 1 张指定范围内的海岸线地图。

```
usamap(mylats,mylons,'line')
```

（6）用 meshm 函数给高程着色，并相应设置颜色查找表。

```
meshm(cape1,caperef1,size(cape1),cape1);
demcmap(cape1)
```

生成的地图如图 41-1 所示，它显示了海角地图的一部分。从图中还可看出，利用 0 级 DTED 数据得到的图形相对而言是比较粗糙的。

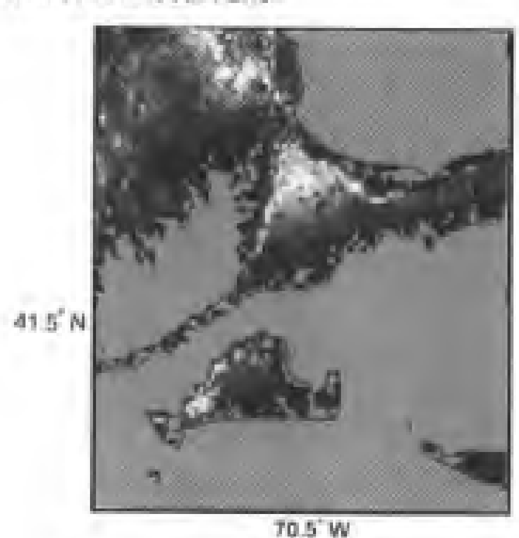


图 41-1　用 0 级 DTED 数据得到的地图

当目标区域超出 1 个 DTED 窗格的范围时，dted 函数会将这些窗格聚合到一个单独的矩阵中。可以指定一个单独的 DTED 文件，一个包含几个文件的目录或包含几个子目录的更高级目录，这些子目录中包含几个经度带文件。

要继续上面的例子，需要从 http://www.nima.mil 获取必要的 DTED 文件。可以获取包含下面文件的系列目录。

```
/dted
  /w070
    n41.avg
    n41.dt0
    n41.max
    n41.min
    n43.avg
    n43.dt0
    n43.max
    n43.min
  /w071
    n41.avg
    n41.dt0
    n41.max
    n41.min
    n42.avg
    n42.dt0
    n42.max
    n42.min
    n43.avg
    n43.dt0
    n43.max
    n43.min
  /w072
    n41.avg
    n41.dt0
    n41.max
    n41.min
    n42.avg
    n42.dt0
    n42.max
    n42.min
    n43.avg
    n43.dt0
    n43.max
    n43.min
```

将工作目录改为包含顶级 DTED 目录（名称总是 dted）的目录。用 dted 函数将该目录指定为第 1 个变量。还必须指定取样间隔、纬度范围限制和经度范围限制作为第 2 个到第 4 个变量

```
[capetopo,caperef] = dted(pwd, 5, [41.1 43.9],[-71.9 -69.1]);
```

取样间隔越大，将要输出的网格文件越小。

因为 DTED 模型不包含海洋深度信息，将所有的 0 高程重新编码为-1，使得水域内也能着上合适的颜色。

```
capetopo(capetopo==0)=-1;
```

然后获取网格的纬度和经度限制，用它们绘制区域的轮廓图。

```
[latlim,lonlim] = limitm(capetopo,caperef);
usamap(latlim,lonlim,'line')
```

用 meshm 函数对高程网格进行着色，然后用 demcmap 函数重新着色，以显示海水的颜色。

```
meshm(capetopo,caperef,size(capetopo),capetopo);
demcmap(capetopo)
```

41.2　交互读取高程数据

使用 demdataui 图形用户界面，可以浏览许多格式的数字高程地图数据。下面的例子用该界面提取 DEM 数据。按照以下步骤进行：

（1）打开 demdataui 图形用户界面，系统会在显示数据之前扫描数据所在的路径。在命令窗口输入

```
demdataui
```

显示图 41-2。

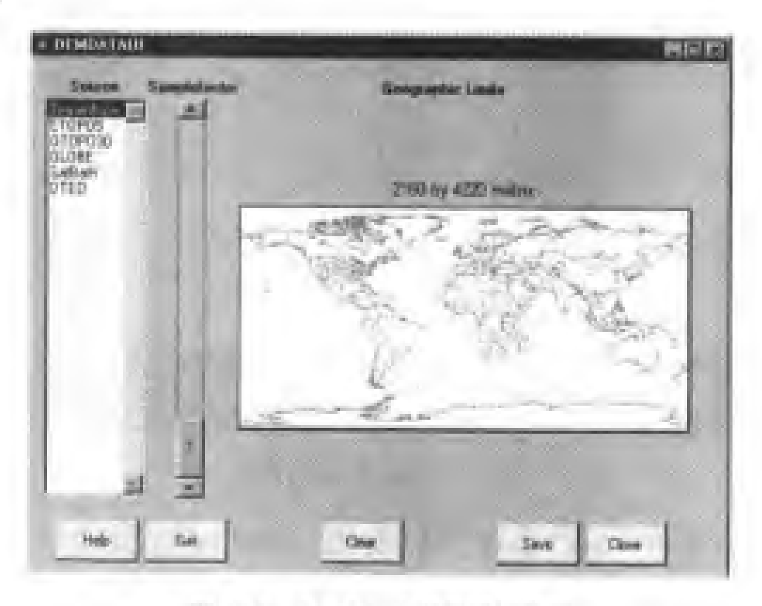


图 41-2　demdataui 图形用户界面

（2）面板左侧的 “Source” 列表框显示了找到的数据集。每套数据覆盖的范围在地图上用黄色表示。

（3）在列表框中单击不同的数据源，右侧的地图中将会自动更新显示。图 41-3 中表示了 SatBath 数据集覆盖的区域。

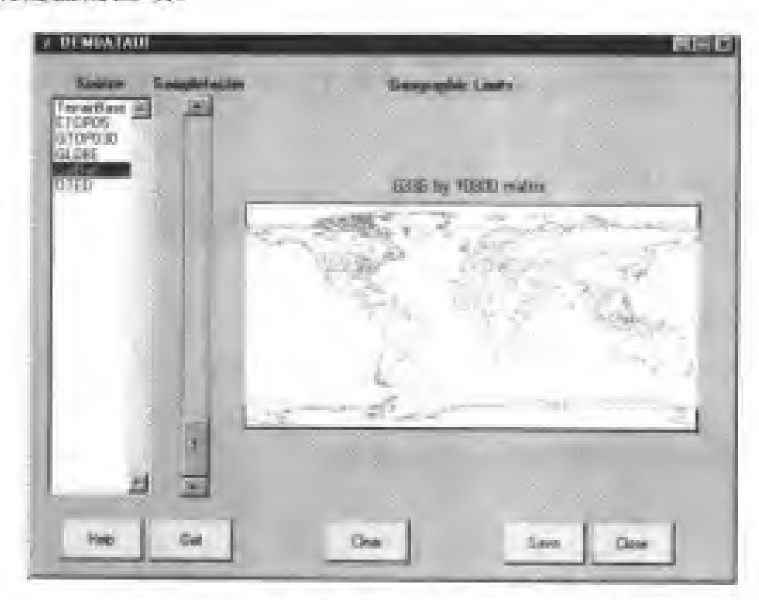


图 41-3　显示 SatBath 数据集覆盖的区域

（4）图 41-4 中的地图用于确定需要提取多少数据。当前显示区域的矩阵所需要的存储量显示在地图上方。要减少数据量，在地图上单击并拖拉可以放大地图。

图 41-4 中是选择 TerrainBase 数据并放大显示印度次大陆以后的显示结果。

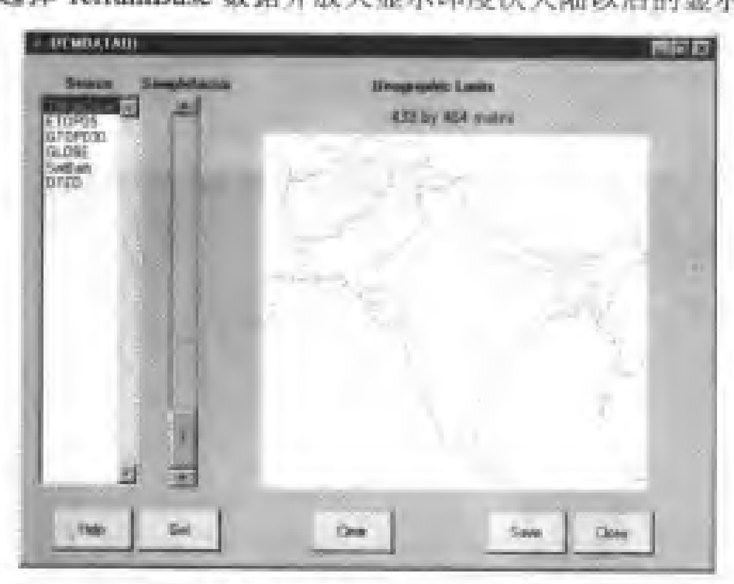


图 41-4　TerrainBase 数据显示

（5）在图 41-4 中单击 “Get” 按钮，显示地形，如图 41-5 所示。

（6）如果对当前结果不满意，单击 “Clear” 按钮，删除所有先前用 “Get” 按钮读入的数据，并选择新数据。



图 41-5　地形图

（7）准备导入 DEM 数据到工作空间或保存为 MAT 文件时，单击“Save”按钮。然后 MATLAB 会提醒你选择输出变量或文件的路径或名称。可以保存为 MAT 文件或保存到工作空间变量。demdataui 函数返回 1 个或多个矩阵，作为一个地理数据结构数组。然后可以用 displaym 或 mlayers 函数添加数据网格到地图坐标系中。

（8）要获得地理数据结构的内容，需要使用它的字段名。下面的代码中，map 和 maplegend 字段从结构中被复制出来，并在用 worldmap 函数创建有光照的三维高程地图时使用。

```
map = demdata.map;
maplegend = demdata.maplegend;
figure
worldmap(map,maplegend,'ldem3d')
hidem(gca)
```

生成图 41-6。

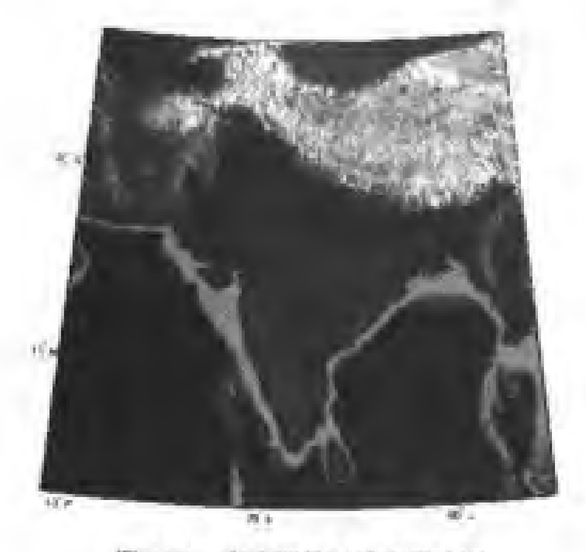


图 41-6　有光照的三维高程地图

41.3　确定整个地形上的可见性并进行显示

可以用高程数据的规则网格回答表面某点可见性的问题，例如，

- 一点到另一点的视线是否被地形遮挡；
- 当前位置可以看到哪些地方；
- 从哪些地方可以看到当前点。

第 1 个问题可以用 los2 函数回答。该函数可以确定表面上两点之间的可见性。

下面的例子演示了规则数据网格上两点之间的视线计算，这个网格用 peaks 函数生成。用 los2 函数进行计算，它返回一个 logical 型结果，为 1 时表示可见，为 0 时表示不可见。

（1）用 peaks 函数创建一个高程网格，最大高程为 500，将它的原点设置为(0°N, 0°W)，间隔为每度 1 000 个单元。

```
map = 500*peaks(100);
maplegend = [ 1000 0 0];
```

（2）在该网格上定义两个点测试它们的可见性。

```
lat1 = -0.027; lon1 = 0.05; lat2 = -0.093; lon2 = 0.042;
```

（3）计算可见性。最后一个变量指定第 1 个位置(lat1, lon1)上表面的高程：

```
los2(map,maplegend,lat1,lon1,lat2,lon2,100)
ans =
1
```

los2 函数还在图形窗口中生成一个参考图，显示视线沿线每个网格单元的可见性。本例中，图 41-7 显示两点之间的视线正好被一个中间的峰给挡住了。

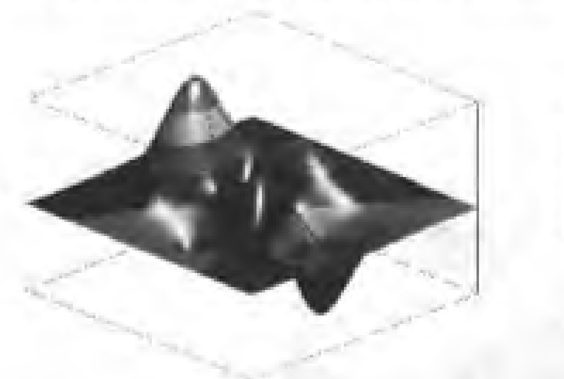
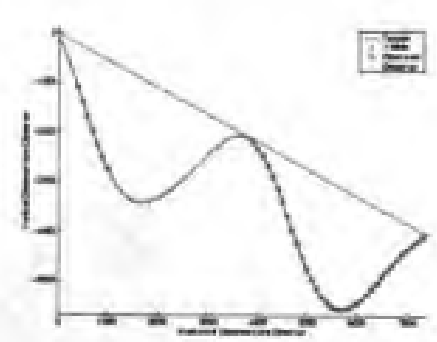



图 41-7　视线可见性示意图

还可以计算分视岭。分视岭源于分水岭这一名称，表示某个特定点上可以看到的所有区域。用 viewshed 函数计算分视岭，该函数支持与 los2 函数相同的选项。

下面显示图 41-7 中站在最高点上可以看见的所有区域，用蓝色表示。

```
[vismap,vismapleg] = viewshed(map,maplegend,lat1,lon1,100);
```

效果如图 41-8 所示。

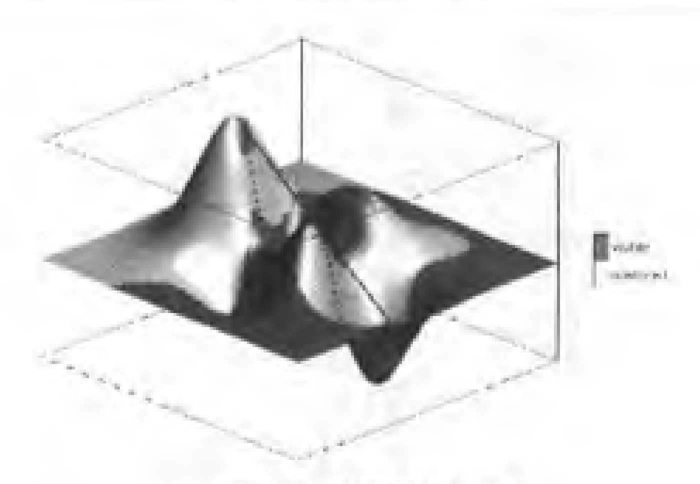


图 41-8 最高点处的视域

41.4 给地形图添加阴影和光照

用 lightm 函数在当前地图中创建 Light 对象。可以用交互式 lightmui GUI 工具修改地图上光照的位置和颜色。为了更好地控制光照位置，必须使用投影坐标进行指定。

41.4.1 给 DTED 文件创建的地形图添加光照

下面的例子将光照的位置手工指定在海角地图 DTED DEM 的西北角。按照下面的步骤进行操作：

（1）运行下面的代码，生成海角的地图。

```
mylats = [41.2 41.95];
mylons = [-70.95 -70.1];
cd dted\w071        % Note: Your absolute path may vary
[cape1,caperef1] = dted('n41.dt0',1,mylats,mylons);
cape1(cape1==0)=-1;
usamap(mylats,mylons,'line')
meshm(cape1,caperef1,size(cape1),cape1);
demcmap(cape1)
```

生成的地图如图 41-9 所示。

（2）设置地形在垂向上的拉伸程度。用 daspectm 函数指定高程的度量单位为米，并且用 20 去乘。

```
daspectm('m',20)
```

（3）确认视线可见。为了保证视线不会被地形遮挡，用 zdatam 函数将视点设置为 cape1 地形数据中高程值的最大值。

```
zdatam('allline',max(cape1(:)))
```

（4）用 lightm 函数指定光源位置。

```
h = lightm(42,-71)
```

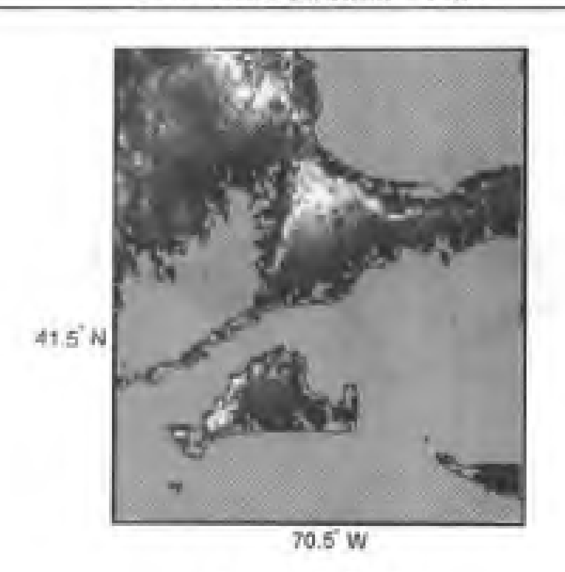


图 41-9　海角地图

如果忽略变量，会打开一个设置新光照位置属性的 GUI。

（5）为了查看光照对象的属性，结合 lightm 函数返回的句柄，用 get 函数进行显示。

```
get(h)
        Position = [ -0.00616097 0.796039 1]
        Color = [1 1 1]
        Style = infinite

        BeingDeleted = off
        ButtonDownFcn =
        Children = []
        Clipping = on
        CreateFcn =
        DeleteFcn =
        BusyAction = queue
        HandleVisibility = on
        HitTest = on
        Interruptible = on
        Parent = [138.001]
        Selected = off
        SelectionHighlight = on
        Tag =
        Type = light
        UIContextMenu = []
        UserData = [ (1 by 1) struct array]
        Visible = on
```

因为已经在使用 lightm 函数时使用了 MATLAB 的 light 函数，所以早应该在笛卡儿三维空间中指定光照的位置。

（6）因为使用了镜面强光，地图颜色显得比较暗。注意通过将 3 个表面反射属性设置到[0 1]范围来恢复亮度。

```
ambient = 0.7; diffuse = 1; specular = 0.6;
material([ambient diffuse specular])
```

因为没有使用插值光照（使用的是刻面光照），所以图形表面看起来有很多斑点。可以通过指定 Phong 方法进行矫正。

现在的地图如图 41-10 所示。

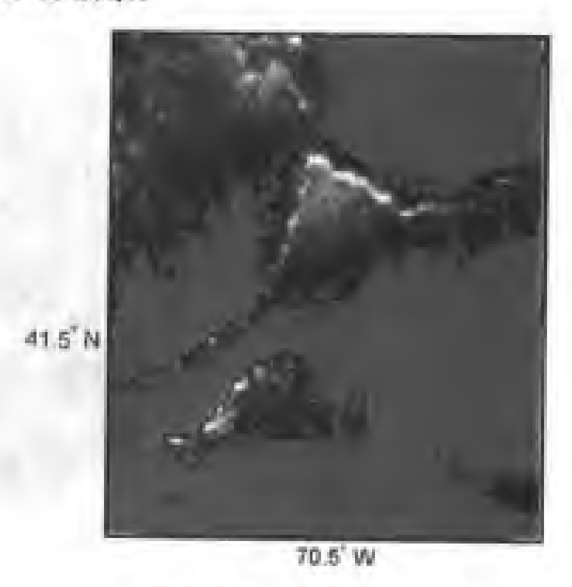


图 41-10　添加光照后的地图

（7）取消光照，可以将有光照和没有光照时的显示效果进行对比。

```
lighting none
```

41.4.2　用 lightm 函数和 lightmui 工具给世界地形图添加光照

下面的例子创建一个世界地形图，并在距离纽约上空 250 公里的地方添加光照。然后改变材质和光照属性，添加第 2 个光源，再激活 lightmui 工具，改变光照的位置、高度和颜色。

lightmui 工具用圆圈表示光对象，圆圈的前景色表示光的颜色。要改变光的位置，单击并拖拉这个圆圈。同样，右击圆圈以后在弹出的对话框中也可以改变光对象的位置或颜色。单击对话框中的色条，会调用 uisetcolor 对话框，可以在该对话框中指定或选择光照的颜色。

（1）载入 topo DTM 文件，并进行正交投影。

```
close all; clear;
load topo
axesm ('mapprojection','ortho', 'origin',[10 -20 0])
```

（2）绘制地形图，指定一个颜色查找表。

```
meshm(topo,topolegend);
demcmap(topo)
```

（3）在纽约上空设置一个黄色光源。

```
hl = lightm(40.75,-73.9, 500/almanac('earth','radius'),...
'color','yellow', 'style', 'local');
```

lightm 函数的前两个变量是光源位置对应的纬度和经度。第 3 个变量是它的海拔高度，单位为地球半径。

（4）地图表面颜色比较暗，所以用下面的代码添加更多的反射成分。

```
material([0.7270   1.5353   1.9860   4.0000   0.9925]);
lighting phong; hidem(gca)
```

现在的世界地形图如图 41-11 所示。



图 41-11　给世界地形图添加光照

（5）还可以添加更多的光照，如下所示。

```
h2 = lightm(20,40, .1,'color','magenta', 'style', 'local')
```

第 2 个光照是洋红色的，在波斯湾的上空：

（6）可以用 lightmui GUI 修改光照，使用它可以在世界地形图上拖拉光对象，并可以指定新的颜色和高度。

```
lightmui(gca)
```

添加的光对象用洋红色的圆圈表示，可以通过拖拉将它移到新位置，也可以在单击圆圈的同时按住<Ctrl>键，然后在弹出的对话框中直接指定光对象的位置、高度和颜色。GUI 和地图如图 41-12 中所示。



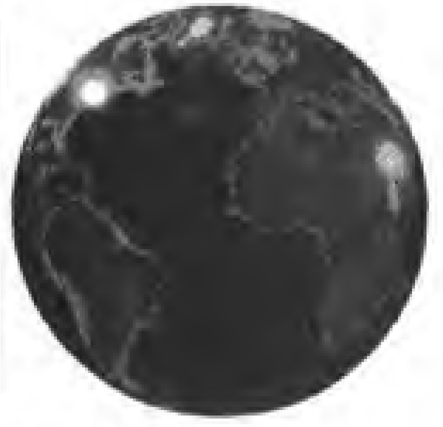


图 41-12　添加两个光照

（7）在 lightmui 窗口中，将黄色光对象拖到巴西最西端，将洋红色的光对象拖到直布罗陀海峡，GUI 和地图如图 41-13 所示。

（8）单击洋红色圆圈的同时按住<Ctrl>键或<Shift>键，然后在弹出的对话框中设置光对象的位置、颜色和开关状况。将高度设置为 0.04（单位为地球半径），颜色设置为（1.0, 0.75, 1.0），单击 Return 按钮，色条上的颜色改变为设置的颜色。如果更喜欢点选颜色，可以单击色条，然后在弹出的颜色选择对话框中进行选择。现在地图如图 41-14 所示。

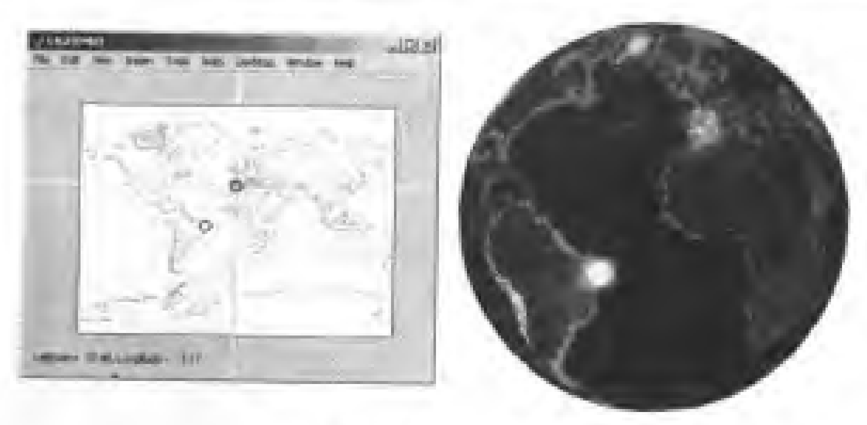


图 41-13　移动光照

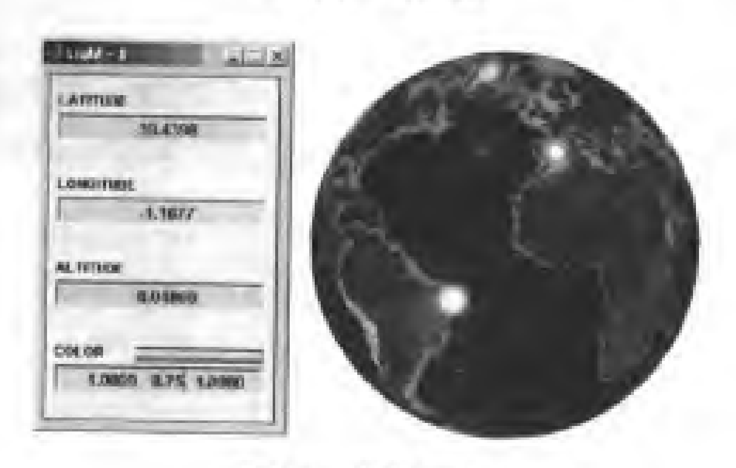


图 41-14　设置光照属性

41.4.3　给地貌添加阴影

可以用 surflm 函数制作空间单色阴影地貌图，该函数与 MATLAB 的 surfl 函数类似。使用 surflm 函数的效果与使用光照的效果差不多，但是函数模型是通过给表面法向加权来获得照亮的效果，而使用光照则需要添加光对象。

如上所述，surflm 函数是通过模拟单个光源而不是插入光对象来进行工作的。下面的例子用 korea 数据集演示了 surflm 函数的使用。这里使用 worldmap 函数设置合适的坐标和参考线。

（1）用 worldmap 函数设置一种投影类型，并显示一幅朝鲜半岛的向量地图。

```
hk = worldmap('korea','lineonly'); framem off;
```

worldmap 函数选择一种投影类型和地图边界来制作该地图，生成的地图如图 41-15 所示。

（2）载入 korea 地形模型。

```
load korea
```

（3）生成纬线和经线网格，将规则数据网格转换为地理定位网格。

```
[klat,klon] = meshgrat(map,maplegend);
```

（4）用 surflm 函数生成一个默认的阴影地貌地图，并将颜色查找表改成单色的，如 gray、bone 或 copper 等。

```
ht = surflm(klat,klon,map);
colormap('copper')
```

默认时，光照方向是相对于视线方向逆时针 5° 的方向，这样，"太阳"就在东南方。现在的地图如图 41-16 所示。

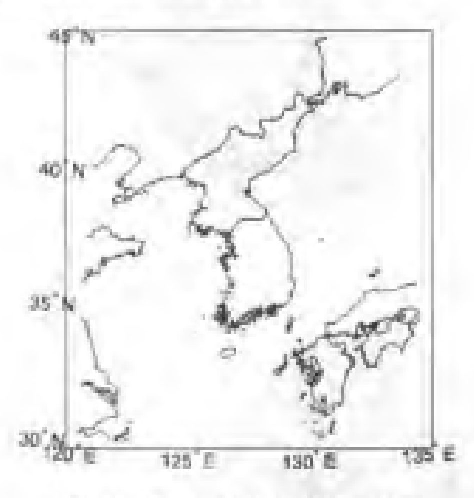


图 41-15　朝鲜半岛的向量地图

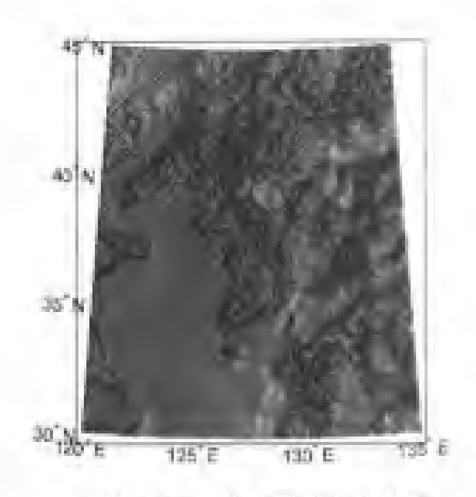


图 41-16　生成阴影地貌地图

（5）要使光线从其他方向射过来，可以将光源的方位角和仰角作为 surflm 函数的第 4 个变量指定。清除地形图并重绘，指定方位角为 135°，仰角为 60°。

```
clmo(ht); ht=surflm(klat,klon,map,[135,60]);
```

地图表面现在更亮了，如图 41-17 中所示。

（6）现在将光线转换到西北（方位角-135°），俯角 40°。

```
clmo(ht); ht=surflm(klat,klon,map,[-135, 30],[.65 .4 .3 10]);
```

对于本例的地形来说，这个设置是一个好的选择，因为大部分山脊的走向都是从北到南，所以多少都与光线的方向平行。图 41-18 是最后生成的地图。

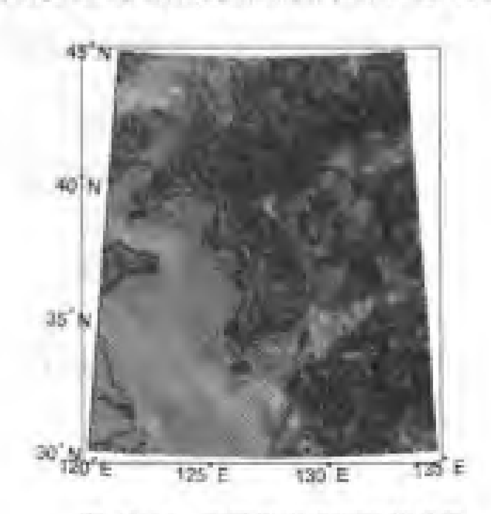


图 41-17　重新指定光源后的效果

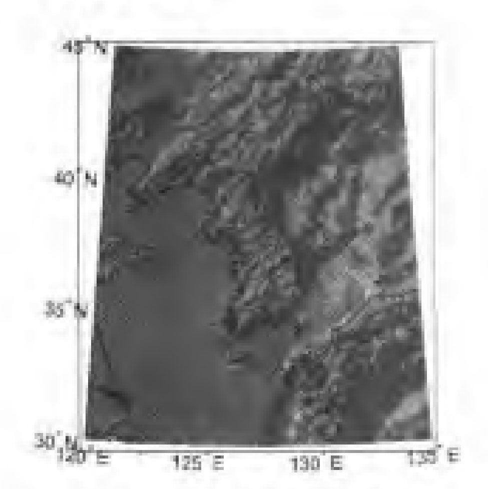


图 41-18　最后生成的地图

函数 meshlsrm 和 surflsrm 通过表面着色和光影把地图显示成阴影地貌图。可以把它们当作 surflm 函数的扩展，它们接合了表面着色和表面光照阴影的特点。用 meshlsrm 函数显

示规则数据网格，用 surflsrm 函数给地理定位数据网格着色。

41.4.4　给阴影地貌图着色并作三维显示

下面的例子介绍 surfsrm 函数的用法。

（1）启动一个新的地图坐标系和 korea 数据，然后使用规则数据网格。

```
close all; clear all;
load korea
[klat,klon] = meshgrat(map,maplegend);
axesm miller
```

（2）给 DEM 数据创建一个颜色查找表，根据指定的太阳方位角和仰角，用 sruflsm 函数进行转换。

```
[cmap,clim] = demcmap(map);
```

（3）指定光源的地平经度为-135°，地平纬度为 50°，绘制彩色的阴影地貌图。

```
surflsrm(klat,klon,map,[-130 50],cmap,clim)
```

使用 meshlsrm 函数也可以达到相同的效果，它基于规则数据网格进行操作，即

```
meshlsrm(map,maplegend)
```

（4）图形表面现在的对比度更强，加亮 25%。

```
brighten(.25)
```

现在，地图的俯视图如图 41-19 所示。



图 41-19　地图的俯视图

（5）图 41-19 是俯视图，下面绘一个斜视图。设置地形地貌的垂向拉伸因子为 50，视图地平经度为-30°，地平纬度为 30°。

```
set(gca,'Box','off')
daspectm('meters',50)
view(-30,30)
```

生成的地图如图 41-20 所示。

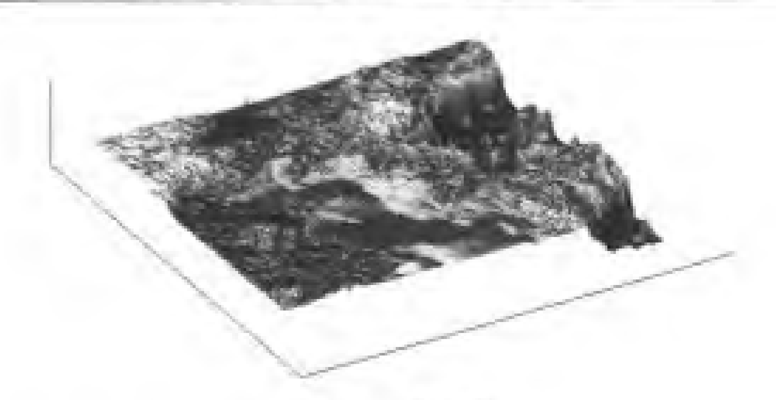


图 41-20　斜视图

41.4.5　用光照对象照亮彩色三维地貌图

作为前面加亮阴影地貌图示例的对比，下面给朝鲜半岛地区的彩色表面图添加光源。

（1）如果需要，载入 korea DEM，并用 Miller 投影创建一个地图坐标系。

```
load korea
figure; axesm('MapProjection','miller',...
        'MapLatLimit',[30 45],'MapLonLimit',[115 135])
```

（2）用 meshm 函数显示该 DEM，然后用地形对应色调进行着色。

```
meshm(map,maplegend,size(map),map);
demcmap(map)
```

没有光照效果的地图如图 41-21 所示。

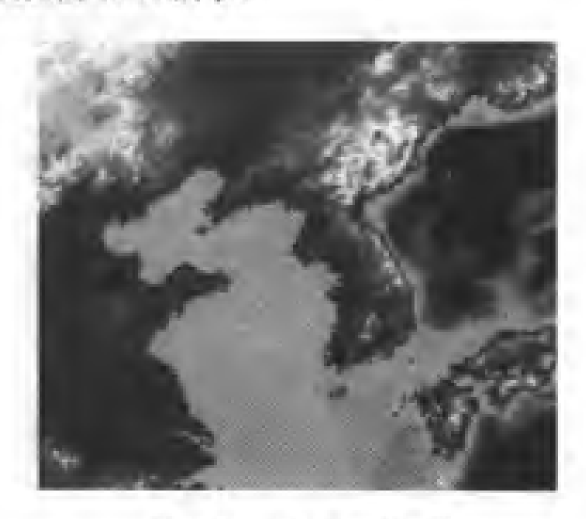


图 41-21　没有光照的地图

（3）用 lightm 函数创建一个光对象。这个光对象放在网格的西北角，地平纬度为 1°。

```
h=lightm(45,115,1)
```

图像现在更暗了。

（4）为了查看地貌的透视效果，用因子 50 拉伸垂向上的显示。

```
daspectm('meters',50)
```

图像仍然变暗，高点处很亮。

（5）分别设置环境光、漫反射光和镜面反射光的反射特性。

```
material ([.7, .9, .8])
```

（6）默认时使用刻面光照。改变为 Phong 光照。

```
lighting phong
```

地图现在如图 41-22 所示。



图 41-22 添加 phong 光照以后的地图

（7）最后，删除包围盒的边，设置地平经度为-30 度，地平纬度为 30 度的视点。

```
set(gca,'Box','off')
view(-30,30)
```

视点在(-30,30)，在(45,115,1)处有一个光对象，进行 Phong 光照以后的显示效果如图 41-23 所示。

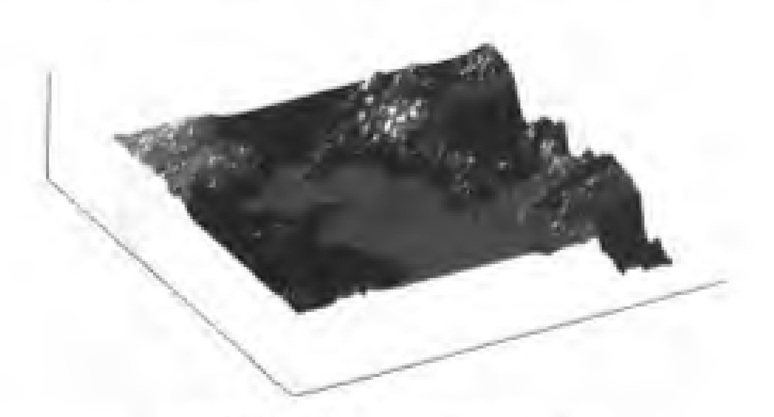


图 41-23 更换视点后的地图显示

从当前图形中删除光照，输入

```
clmo(handlem('light'))
```

41.5 在高程地图上叠加数据

当高程和图像数据网格与相同的地理位置逐像素对应时，可以用表面显示函数中可选的高度参数进行显示。如果不对应，可以将源网格中的一个或两个插值到通用网格中。

41.5.1 在地形图上叠加大地水准面高度

下面的例子在 topo 数据集显示的地形地貌图上叠加 geoid 数据集。两个数据集都是具有公共原点，大小为 1×1 度的网格。按照下面的步骤进行操作。

（1）载入 topo 和 geoid 规则数据网格。

```
load topo
load geoid
```

（2）用 Gall 立体柱面投影（一种透视投影）创建地图坐标。

```
axesm gstereo
```

（3）用 meshm 函数绘制 geoid 变化的彩色表示，将 topo 作为最后一个变量，使 geoid 网格单元的高度与对应 topo 网格单元的相同。

```
meshm(geoid,geoidlegend,size(geoid),topo)
```

geoid 高度较低时用蓝色表示，较高时用红色表示。

（4）为了进行参照，将世界海岸线用黑色表示，高度抬升 1000m，并放大地图，使之充满图框。

```
load coast
plotm(lat,long,'k')
zdatam(handlem('allline'),1000)
tightmap
```

现在，地图如图 41-24 所示。

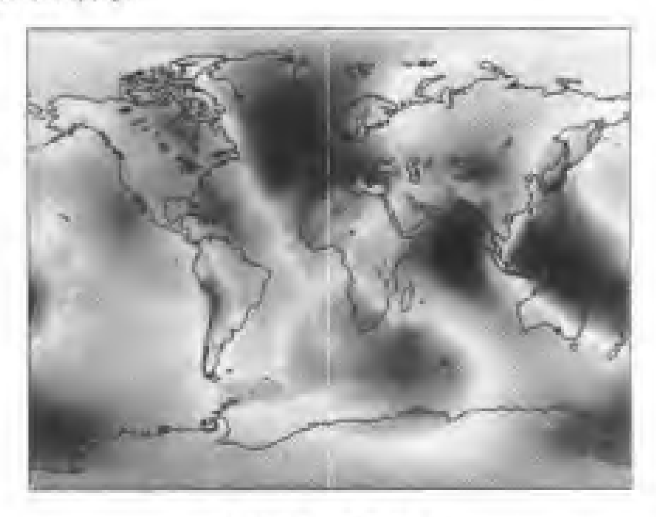


图 41-24　地图效果

（5）因为是俯视图，并且没有光照，所以图中地形地貌还看不到。将地形地貌作垂向拉伸，然后设置视点方位为南-东南方。

```
daspectm('m',200); tightmap
view(20,35)
```

（6）删除包围盒，设置一个光源，然后用 Phong 光照重新着色。

```
set(gca,'Box','off')
camlight;
lighting phong
```

（7）最后，设置投影方式为透视投影。

```
set(gca,'projection','perspective')
```

最后生成的地图如图 41-25 所示。从图中可以看出，geoid 数据集反映出了主要山脉的地形。



图 41-25　最后生成的地图

41.5.2　在地形图上叠加不同的网格数据

如何想组合覆盖同一区域，但是网格化规格不同的高程和属性数据，必须对其中一个矩阵进行重新采样，使得二者一致。如果其中至少有一套网格是地理定位数据网格就太好了，因为使用它们的水平坐标可以利用 ltln2val 函数进行重采样。要组合不相似的网格，可以创建规则数据网格值的地理定位网格版本或创建地理定位数据网格值的规则网格版本。

下面两个例子演示了这样一些方法。

1．通过将规则网格转换为地理定位网格进行叠加

下面的例子将一套源于规则数据网格的坡度数据叠加到源于地理定位数据网格的高程数据，使用的方法适用于所有不相似的网格。本例将地理定位数据作为表面高程的数据源，并将规则数据网格转换为坡度数据。然后该数据被采样，以与地理定位数据网格保持一致，还进行了颜色编码，进行表面显示。

注意，使用 ltln2val 函数在不规则区域重采样规则数据网格时很重要的一点是，规则化网格化数据必须完全覆盖地理定位数据网格区域。

（1）载入地理定位数据网格。

```
load mapmtx lt1
load mapmtx lg1
load mapmtx map2
```

（2）载入 topo 地球规则化数据网格。

```
load topo
```

（3）计算表面显示比率、坡度和 topo 的梯度。

```
[aspect,slope,gradN,gradE] = gradientm(topo,topolegend);
```

（4）用 ltln2val 函数将坡度值插值为地理定位网格。该函数的两个参数 lt1 和 lg1 指定。

```
slope1 = ltln2val(slope,topolegend,lt1,lg1);
```

输出是一个与 map1 的变量范围相匹配的 50×50 的高程网格。

（5）新建一个图形窗口，投影方法为 Miller 投影，然后用 surfm 函数显示坡度数据。

```
figure; axesm miller
surfm(lt1,lg1,slope1,map1)
```

该图主要突出陡峭的悬崖，用于描述高峰、大陆架和海沟等很合适。

（6）颜色表示斜坡的陡缓程度。改变颜色查找表，使最陡的斜坡用洋红色表示，较陡的斜坡用深蓝色表示，平坦的地方用浅蓝色表示。

```
colormap cool;
```

（7）用 view 函数，从较低的视点出发，获取东南方向表面的透视图。

```
view(20,30); daspectm('m',200)
```

三维条件下，可以直接观察到地形和斜坡。

（8）默认时使用刻面光照，改用 Phong 光照并从东面照过来。

```
material shiny;camlight;lighting phong
```

（9）最后，删除空白区域并以透视模式重新对图形进行着色。

```
axis tight; set(gca,'Projection','Perspective')
```

绘图结果如图 41-26 所示。

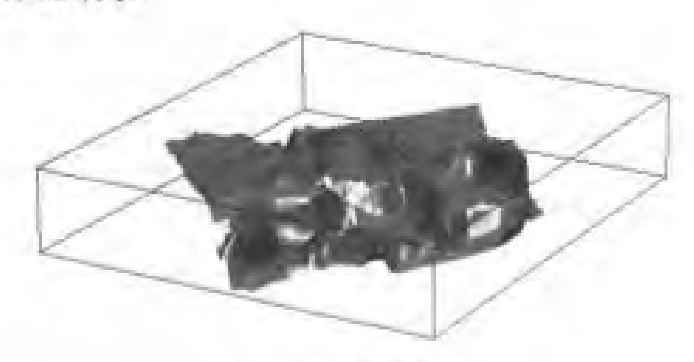


图 41-26　生成的地图

2．通过纹理映射将地理定位网格叠加到规则数据网格上

组合规则网格和地理定位数据网格的第 2 种方法是创建地理定位数据网格 z 数据的规则数据网格。这种方法的好处是有更多计算能力更强的函数更适合于规则数据网格。另一个方面是，颜色和高程网格不必大小相同。如果两张图的精度不同，则可以把表面创建成三维高程地图并在以后将其颜色作为纹理进行映射。这一点可以通过将表面对象的 Cdata 属性设置为包含颜色矩阵，将 Facecolor 属性设置为'TextureMap'来实现。

按照下面的步骤进行操作，可以创建一个新的规则数据网格，它覆盖了地理定位数据网格的区域，然后将颜色数据嵌入新矩阵。这个新矩阵的分辨率可能比原矩阵低一点，以保证新地图中的每个单元接收一个值。

（1）清除工作空间，载入 topo 和地形数据。

```
clear; load topo;
load mapmtx lt1
load mapmtx lg1
load mapmtx map2
```

（2）识别 mapmtx 地理定位网格之一的地理限制。

```
latlim = [min(lt1(:)) max(lt1(:))];
lonlim = [min(lg1(:)) max(lg1(:))];
```

（3）将 topo 数据调整到包含更小网格的矩形区域。

```
[topo1,topo1ref] = maptrims(topo,topolegend,latlim,lonlim);
```

（4）创建一个规则网格，网格用 NaNs 填充，以接收纹理数据。

```
[curve1,curve1ref] = nanm(latlim,lonlim,.5);
```

（5）用 imbedm 函数将源于 map1 的值嵌入到 curve1 网格，这些值是离散拉普拉斯变换的结果。

```
curve1 = imbedm(lt1,lg1,del2(map1),curve1,curve1ref);
```

（6）新建图形窗口，使用 Miller 投影。用 meshm 函数绘制 topo1 数据。

```
figure; axesm miller
h = meshm(topo1,topo1ref,size(topo1),topo1);
```

（7）将图形渲染成三维视图，视点为地平经度 20°，地平纬度 30°，地形垂直拉伸因子为 200。

```
view(20,30); daspectm('m',200)
```

（8）加亮视图，使用 Phong 光照。

```
material shiny; camlight; lighting phong
axis tight; set(gca,'Projection','Perspective')
```

所以，表面地貌和颜色都表示地形高度，地图外观如图 41-27 所示。

（9）现在，使用 set 函数将 curve1 矩阵作为纹理直接映射到图形上。

```
set(h,'Cdata',curve1,'FaceColor','TextureMap')
```

映射结果如图 41-28 所示。

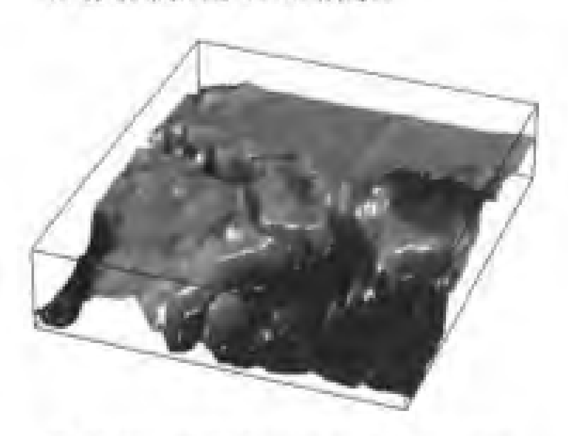


图 41-27　添加 phong 光照后的三维地图

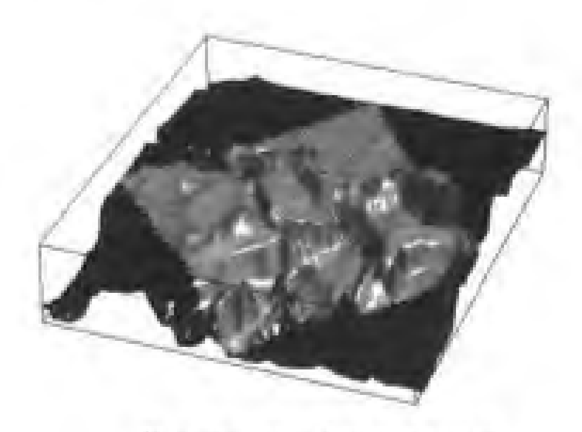


图 41-28　给三维地图映射纹理

41.6　球体显示操作

球体显示是地理空间数据的一种三维视图效果，利用它，可以模拟从空中观察整个星球的地形地貌。通过转换，它将纬度、经度和高程映射到一个三维笛卡儿坐标内。工具箱中的所有投影都将纬度和经度转换成 x 坐标值和 y 坐标值。globe 函数比较特殊，因为它可以在地表垂直方向上进行着色。在原点在地心的笛卡儿坐标系中，z 不是可选的高程，而是笛卡儿三维空间中的一个坐标轴。globe 函数对于需要使对象间三维关系保持不变的地理空间

应用很有用。这样的应用有飞越模拟、观察星球的旋转等。

球体显示是基于坐标变换而不是地图投影的。为了给图形窗口中的球体进行着色，必须应用透视变换或正交变换。这两个变换都涉及到视点设置、背面隐藏和形状、比例和角度变化等问题。

41.6.1　在球体显示中使用透明性

因为球体显示描绘的是三维对象，没有不透明表面遮挡时可以看到对象内部甚至穿过对象看到对象的另一侧。这样可能会使图像有些混乱，因为地球背面的地物影像叠加到前面的地物影像上了。

下面创建一个可以显示直线和点数据的表面：

（1）新建一个图形窗口，进行球体显示。

```
figure; axesm('globe')
```

（2）用浅色绘制经线和纬线。

```
gridm('GLineStyle','-','Gcolor',[.8 .7 .6],'Galtitude', .02)
```

（3）载入 coast 数据并用黑色绘图，进行三维透视。

```
load coast
plot3m(lat,long,.01,'k')
view(3)
```

现在三维视图效果如图 41-29 所示。

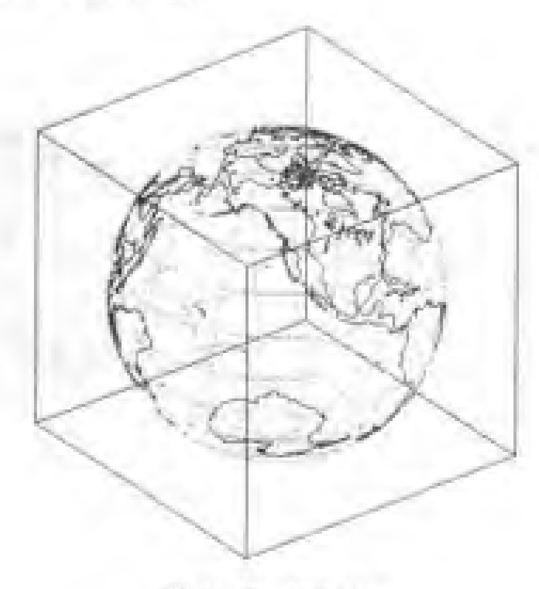


图 41-29　球体显示

（4）用图形窗口工具条中的"Rotate 3D"工具旋转视图。可以发现，由于球体是透明的，图形显得比较杂乱。

（5）生成均匀间隔的 1×1 度网格和一个参考向量。

```
base = zeros(180,360); baseref = [1 90 0];
```

（6）将网格叠加到球面上，用紫铜色对表面着色，在右侧进行光照，使表面更亮。

```
hs = meshm(base,baseref,size(base));
colormap copper
camlight right
material([.8 .9 .4])
```

生成图 41-30。

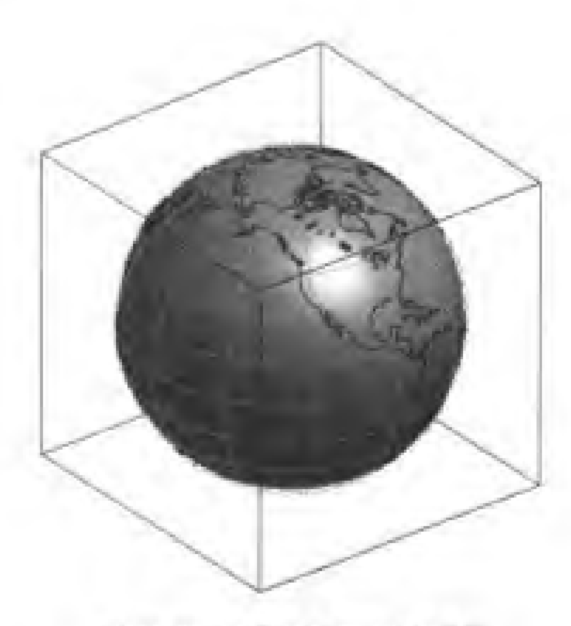


图 41-30　重新设置后的球体显示

41.6.2　用相机定位函数进行水平三维视图

可以用球体显示创建生动的三维视图。利用 camtargm 和 camposm 函数，可以在地理坐标中分别设置焦点和视点的位置，所以不需要处理三维笛卡儿图形坐标。

下面的练习中，将 worldlo 数据集定义的国界叠加到地形地貌图上，然后从华盛顿上空向莫斯科方向观察地球。

（1）创建球体显示，并获取地形数据。

```
clear; axesm globe
load topo
```

（2）忽略 meshm 函数的第 4 个变量，显示 topo 数据集时不显示垂向组分。

```
hs = meshm(topo,topolegend,size(topo)); demcmap(topo);
```

默认视图是从北极上空往下看，中央子午线与 x 轴平行。

（3）用 displaym 函数将 worldlo 数据集中 POline 地理结构内的国界图叠加到上面生成的图中，国界用亮灰色显示。

```
hl = displaym(worldlo('POline')); set(hl,'color',[.7 .7 .7])
```

（4）用 extractm 函数从 worldlo 数据集的 gazetteer 结构中找到莫斯科和华盛顿的坐标位置。

```
[tlat,tlon] = extractm(worldlo('gazette'),'Moscow');
[plat,plon] = extractm(worldlo('gazette'),'Washington');
```

（5）创建连接华盛顿和莫斯科的大圆，并用红色绘出来。

```
[latc,lonc] = track2('gc',tlat,tlon,plat,plon);
plotm(latc,lonc,'r')
```

（6）将相机放在莫斯科。不管以后相机移到哪里，它总是对准[tlat, tlon]的。

```
camtargm(tlat,tlon,0)
```

（7）将相机放在[plat,plon]点上，第 3 个变量是地球半径表示的高度。

```
camposm(plat,plon,3)
```

（8）用相机目标点的坐标建立相机抬升向量。现在连接华盛顿和莫斯科的大圆在垂直方向上。

```
camupm(tlat,tlon)
```

（9）将相机的视野设置为 20°。

```
camupm(20)
```

（10）添加光照，指定一个相对弱反射的表面材质，并进行消隐处理。

```
camlight; material(0.6*[ 1 1 1])
hidem(gca)
```

最终生成图 41-31。

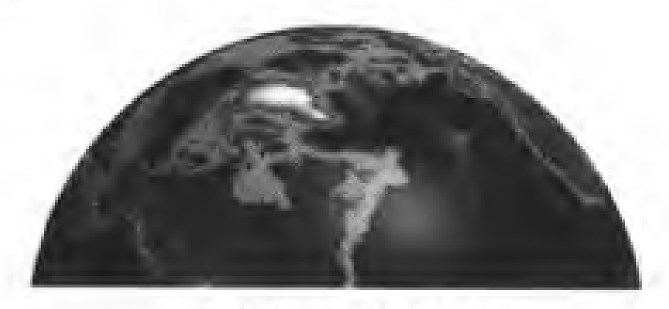


图 41-31　最终生成的地图

41.6.3　显示一个旋转的地球

因为从各个方向观察球体时不需要重新计算投影，所以可以比较容易地用动画来表现一个旋转的地球。如果显示的数据足够简单，这个动画就可以以相对较快的速度进行绘制。下面的例子中，首先用一个 M 文件设置地球从西向东按照一度的增量旋转时如何重新设置视图，然后利用该 M 文件控制地球旋转时的显示效果。

（1）在 MATLAB 编辑器中创建一个包含下面代码的 M 文件。

```
for i=360:-5:0
    view(i,0);
    drawnow
end
```

（2）将它保存到当前路径或 MATLAB 路径中，名称为 spin.m。注意，图像的方位角参数与地理方位角的不同：西面为 90°。

（3）用格子线创建球体显示效果，如下所示。

```
axesm('globe','Grid','on','Gcolor',[.7 .8 .9],'GlineStyle','-')
```

视图方向是北极上方。

（4）隐藏图形包围盒的边线，进行透视投影。

```
set(gca, 'Box','off', 'Projection','perspective')
```

（5）用 M 文件旋转球体。

```
spin
```

地球快速旋转，最后位置如图 41-32 所示。

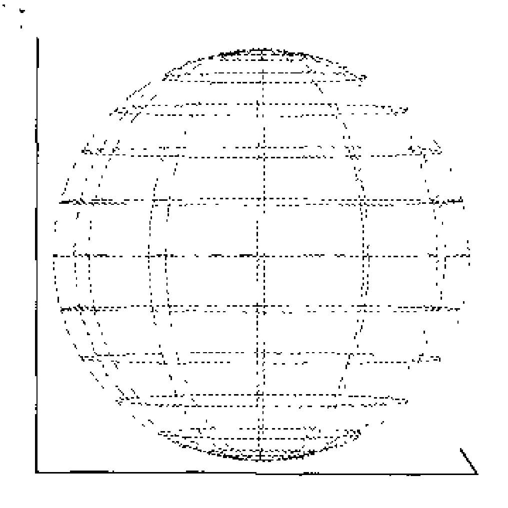


图 41-32　旋转球体

（6）创建海平面数据网格，使球体不透明。

```
base = zeros(180,360); baseref = [1 90 0];
hs = meshm(base,baseref,size(base));
colormap copper
```

现在球体呈均匀的深铜色，叠加了网格。

（7）抬升网格，使它浮在球面上方 2.5%半径距离的地方。

```
setm(gca, 'Galtitude',0.025)
```

（8）再次旋转地球。

```
spin
```

现在旋转慢多了，因为每次旋转都要对这 180×360 的网格进行着色，如图 41-33 所示。

（9）准备用地形地貌数据替换目前的球面数据。

```
clmo(hs)
load topo
```

（10）将高程放大 50 倍显示，绘制表面。

```
topo = topo / (almanac('earth','radius')* 20);
hs = meshm(topo,topolegend,size(topo),topo);
demcmap(topo)
```

（11）再次旋转。

```
spin
```

效果如图 41-34 所示。

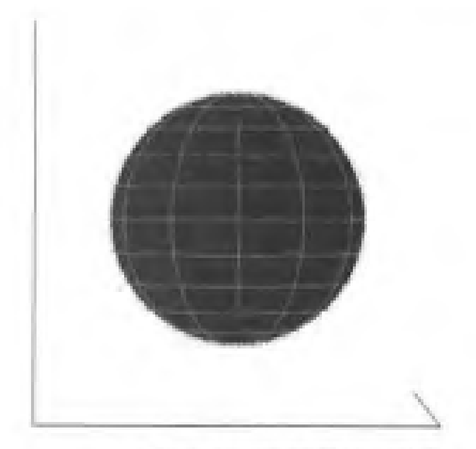


图 41-33　对球面进行着色



图 41-34　添加地形地貌

（12）也可以应用光照，它将跟随星球一起旋转。

```
camlight right
lighting phong;
material ([.7, .9, .8])
```

如图 41-35 所示。



图 41-35　添加光照

第42章 定制地图

使用地图制作工具箱，可以定义各种注解元素和制作各种类型的主题图。

42.1 插入地图

插入地图技巧常常用于以相同比例尺显示单独分开的区域，或者用更小的比例尺显示地图轮廓。可以通过在一个图形窗口中嵌套多个坐标系并为每个坐标系定义合适的地图投影来创建插入地图。为了保证所有地图的比例尺相同，用 axesscale 函数改变它们的大小。下面举一个例子，在南美洲地图上以相同比例尺插入加利福尼亚州的地图。

（1）用 worldmap 函数定义南美洲的地图。

```
close all; clear all; h1 = worldmap('south america');
setm(h1,'FFaceColor','w')  % set the frame fill to white
```

（2）将坐标系放在下面中间的位置上，投影加利福尼亚州的线形轮廓图。

```
h2 = axes('pos',[.5 .2 .1 .1]);
usamap('californiaonly','lineonly')
```

（3）设置图框和标签。

```
setm(h2,'FFaceColor','w')
mlabel; plabel; gridm     % toggle off
```

（4）将子坐标系的比例尺设置为与父坐标系的相同，隐藏图边。

```
axesscale(h1)
hidem([h1 h2])
```

生成图 42-1。

注意，对于每个区域，根据位置和形状的不同，工具箱会选择不同的投影方法和合适的参数。可以覆盖这些选项，使得两个投影相同。

（5）找出地图使用的投影方法，然后使南美洲的投影方法与加利福尼亚州的相同。

```
getm(h1, 'mapprojection')
ans =
  eqdconic
getm(h2, 'mapprojection')
ans =
  lambert
setm(h1, 'mapprojection', getm(h2, 'mapprojection'))
```

（6）最后，通过改变插入地图的属性如颜色等进行试验。

```
setm(h2, 'ffacecolor', 'y')
```

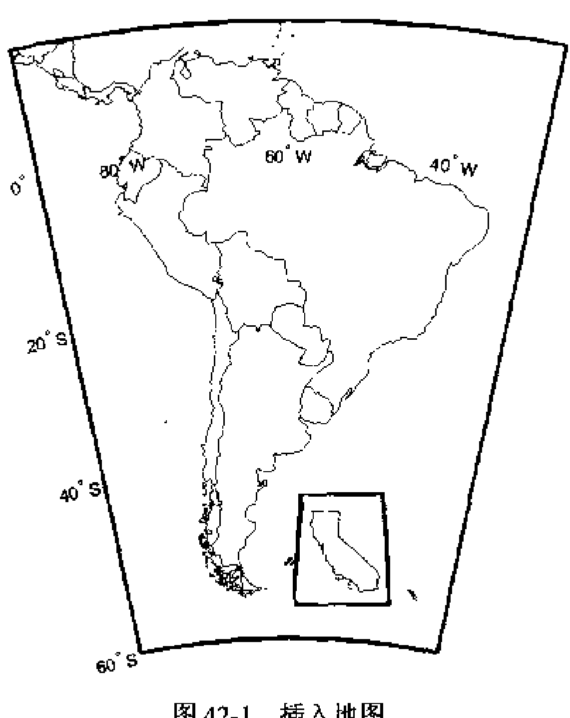


图 42-1　插入地图

42.2　图形比例尺

比例尺更多地是用于确定地图上地物的大小比例。可以用 scaleruler 函数将图形比例尺添加到当前地图。可以用 getm 和 setm 函数检查和修改 scaleruler 设置。也可以通过拖拉比例尺的基准线将它移动到新的位置。

下面的例子创建一幅地图，在地图上添加比例尺，并移动比例尺的位置。然后添加一个单位为海里的比例尺，并改变比例尺的标注样式和方向。

（1）绘制危地马拉的面片图。

```
clear all; close all;
worldmap('lo','guatemala','patchonly')
```

（2）添加默认的图形比例尺，然后将它移动到更高的位置。

```
scaleruler
setm(handlem('scaleruler1'),'YLoc',.205)
```

（3）在地图上放置第 2 个图形比例尺，如果不进行手工调整，它看起来是错的。

```
scaleruler('units','nm')
setm(handlem('scaleruler2'),'MajorTick',0:100:300,...
    'MinorTick',0:25:50,'TickDir','down',...
    'MajorTickLength',km2nm(25),...
    'MinorTickLength',km2nm(12.5))
```

（4）试用两个不同的比例尺样式。

```
setm(handlem('scaleruler1'),'RulerStyle','lines')
setm(handlem('scaleruler2'),'RulerStyle','patches')
```

如图42-2中所示。

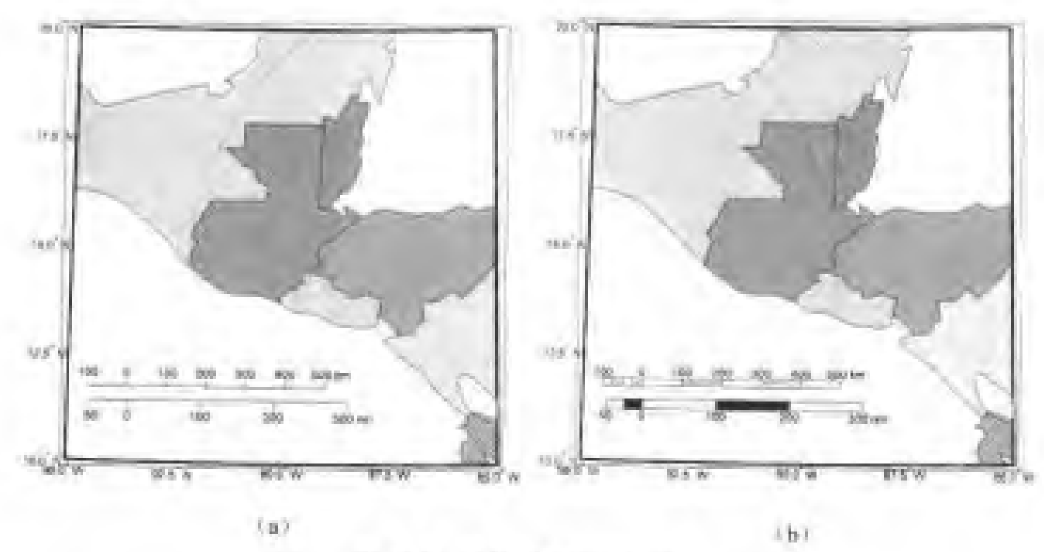


图42-2　给地图添加比例尺

42.3　指北针

指北针指向地理北极，提供了地图上的方向参照。可以用northarrow函数在当前地图上显示一个指北针。单击并进行拖拉，可以移动指北针。移动时会自动计算指北针的方向，不需要进行手工调整。按下Ctrl键的同时单击图标会弹出一个输入对话框，利用它可以改变指北针的位置。

（1）为了演示指北针的使用，创建一幅南极图像并在指定的地理位置添加一个指北针。

```
close all; clear all;
figure; worldmap('south pole')
northarrow('latitude',-57,'longitude',135);
```

（2）单击指北针的光标并将它拖拉到地图的另一角。注意，它始终指向北极。

（3）将指北针拖回到左上角。

（4）右击或按下Ctrl键的同时单击指北针，显示"Inputs for North Arrow"对话框。利用该对话框指定直线宽度、颜色和箭头的相对大小等。将LineWidth属性设置为2并单击OK按钮。图42-3是地图现在的外观。

（5）手工设置某些指北针属性。

```
h = handlem('NorthArrow');
set(h,'FaceColor',[1.000 0.8431 0.0000],...
'EdgeColor',[0.0100 0.0100 0.9000])
```

（6）再创建3个指北针，用于显示南极上每个方向都是北方。

```
northarrow('latitude',-57,'longitude',45)
northarrow('latitude',-57,'longitude',225)
northarrow('latitude',-57,'longitude',315)
```

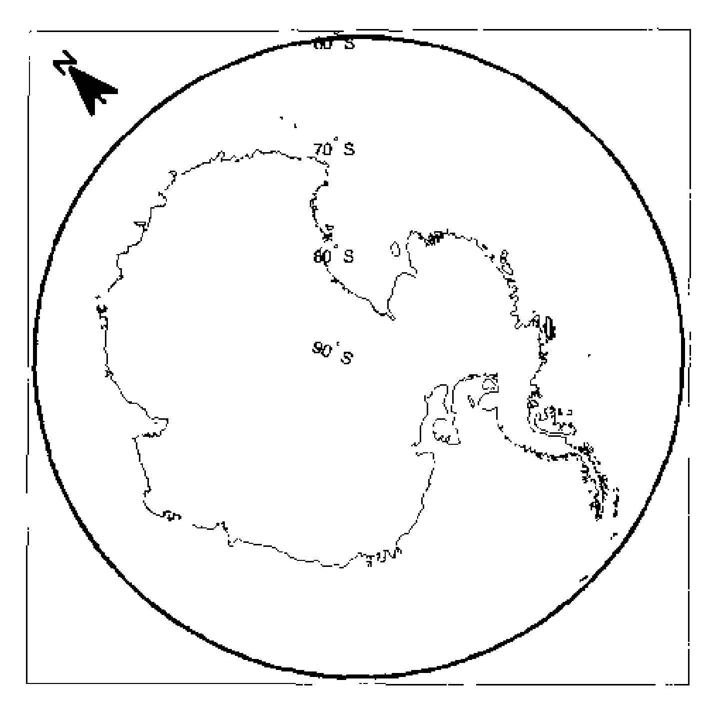


图 42-3　给地图添加指北针

42.4　主题图

主题图不仅仅显示地表的物理特性，如海岸线、道路、居民点、地形和植被等，还可以显示定量特性，如某一区域或多个区域的数据统计量。与主题图有关的有很多制图标记方面的专业词汇，如点标记、点圆分布、向量图、等值线图、色区图、凸棱柱图和连续三维表面等。

42.4.1　地区分布图

最常见的主题图大概是地区分布图。通常在报纸、杂志和报告中用于表示数据。地区分布图用颜色或花纹填充地理分区，表示各种数据值。因为不同数据的个数常常要大于可以提供的标记或颜色个数，地区分布图通常会将它们的数据分成不同的范围。

地图制作工具箱使用面片对象创建地区分布图。它给每个面片表面指定一种颜色，表示特定的变量，每个面片一个值。当变量是标量时，通常表示密度、强度或偶发率。

制作分布图，需要给每个面片输入或计算一个值向量。可直接用地图制作工具箱进行数据值的标记化。这需要到将数据值赋给系列面片的 CData 属性，然后用合适的颜色框架和范围建立颜色查找表。颜色查找表通常将 N 种或更少的值映射给 M 种颜色。M 可以是 2 与 N 之间的任何数字，但主要还是界于 5 和 10 之间。

下面的例子中，用 areaint 函数计算表示美国 50 个州的面片的面积，然后用得到的面积进行显示和着色。对于本例，使用等面积投影比较合适。

（1）载入美国各州的面片数据：

```
close all; clear all; load usalo
```

该数据集包括美国各个州和五大连湖的面片数据。

（2）用合适的投影方法创建地图坐标系。

```
axesm('MapProjection','eqaconic','MapParallels',[],...
MapLatLimit',[15 75],'MapLonLimit',[-175 -60],...
'MLineLocation',15,'MLabelParallel','south',...
MeridianLabel','on','ParallelLabel','on',...
GLineStyle','-','GColor',0.5*[1 1 1],...
Grid','on','Frame','on')
```

（3）绘制 state 结构中的多边形面片地图。

```
displaym(state)
```

（4）指定给面片的颜色基于默认的颜色查找表和面片排序，面片按字母顺序排序。可以用下面的命令行进行查看。

```
tags = {state.tag}
tags =
  Columns 1 through 5
    'Alabama'    'Alaska'    'Arizona'    'Arkansas'    'California'
...
```

（5）选择一个计算球面面积的椭球体。

```
refvec = almanac('earth', 'geoid');
```

（6）用一个 for 循环计算美国所有州的面积。

```
maxarea = 0.0;
for i=1:length(state)
    at = state(i).lat;
    long = state(i).long;
    surfarea = sum(areaint(lat, long, refvec));
    set(handlem(tags{i}),'CData', surfarea);
    maxarea = max(surfarea, maxarea);
end
```

（7）设置颜色查找表中值间隔的范围。

```
caxis([0 maxarea])
```

（8）显示一个色条。

```
colorbar
```

（9）选择一个颜色查找表。

```
colormap('autumn')
```

（10）地图大部分是红的，如图 42-4 所示。可以用其他颜色查找表重新绘制。

注意，因为颜色比例是连续变化的，很多州看起来具有相同的颜色。这是各州面积的不均匀分布造成的。

（11）用下面的命令重新设置 z 轴，将颜色查找表改成色调更多，色阶数目更少的 cool 表，重新绘制色条，显示新的值范围：

```
caxis([10000 1000000])
colormap(cool(16))
colorbar
```

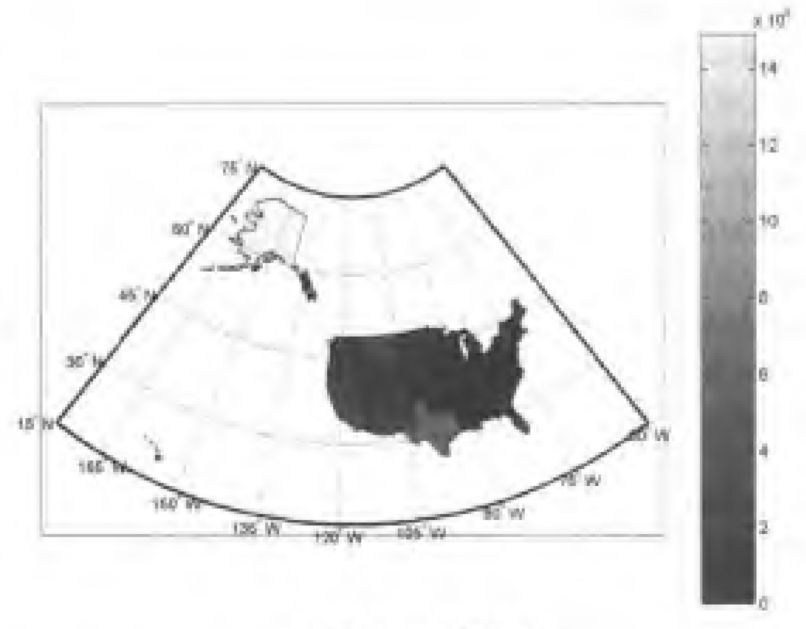


图 42-4　地区分布图

生成的地图如图 42-5 所示。

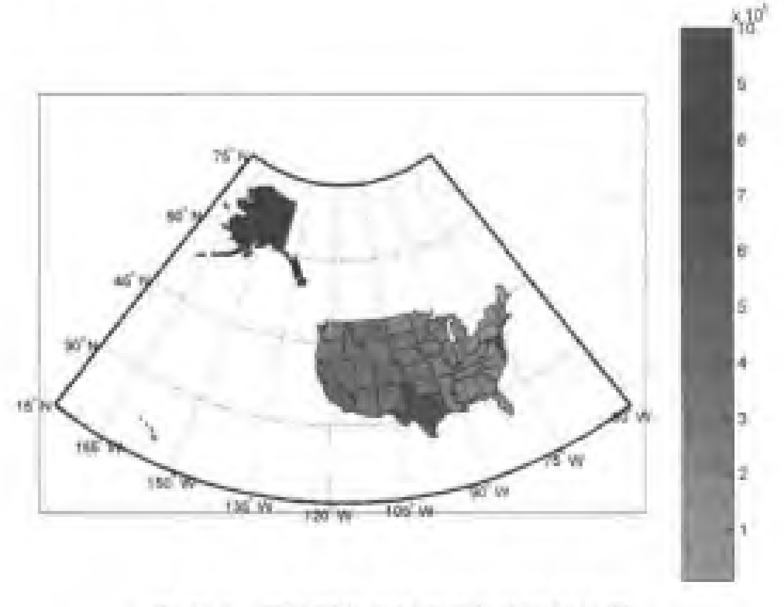


图 42-5　更改颜色查找表后的地区分布图

42.4.2　杆状图

杆状图是描述点上数值分布的三维地理条形图，通常显示在向量基准地图上。图 42-6 是一个杆状图的例子，地图是美国大陆地图。

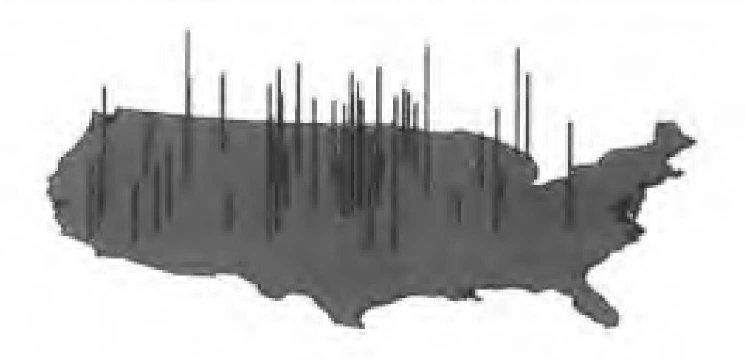


图42-6 杆状图

42.4.3 等值线图

等值线图和后面的向量图分析矩阵数据时很有用。图 42-7 所示的例子中，在地形图上绘制了高程等高线。



图42-7 等值线图

42.4.4 散点图

用 scatterm 函数绘制指定点上的标记。如果标记很小，并且大小没有变化，则生成点分布图。如果标记大小和形状根据分布值的向量有变化，结果是生成比例标记地图。

下面用 scatterm 函数创建北方的星空图。星星用填充圆表示，其大小与可见大小成比例。运行下面的命令行，生成图 42-8。

```
close all; clear all
load stars
index = find(vmag <= 0);
vmag(index) = eps;
axesm('MapProjection','ortho','Origin',[90 0])
setm(gca,'FLatLimit',[90 0],'MapLatLimit',[90 0])
gridm on
```

```
setm(gca,'LabelFormat','compass','LabelRotation','on')
setm(gca,'MLabelParallel',0,'PLabelMeridian',0)
setm(gca,'MeridianLabel','on','ParallelLabel','on')
setm(gca,'GlineStyle','-')
scatterm(lat, long, vmag, 'b', 'filled')
```

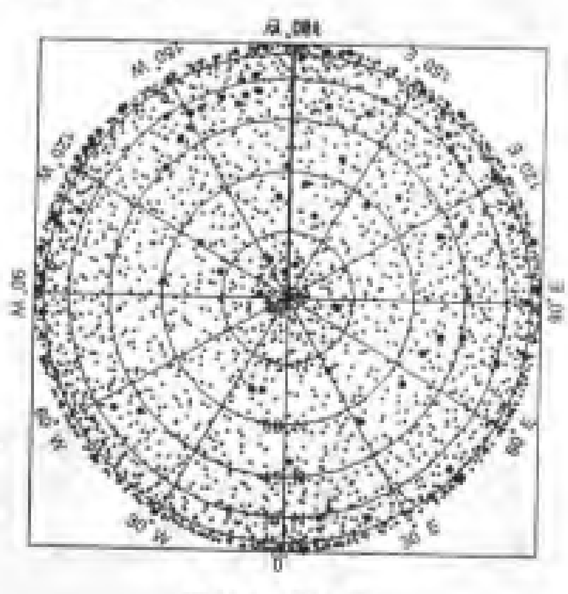


图 42-8　散点图

42.4.5　三角化数据点

地图制作工具箱中没有一个函数可以显示随机数据点的三角形表面。但是，MATLAB 有一个函数可以创建 Delaunay 三角形，下面演示对一些点数据进行三角化，然后将结果放到地图制作工具箱中。

（1）载入 seamount 数据。

```
clear all; close all; load seamount
```

（2）确定坐标的边界，添加 1° 作为空白。

```
latlim = [min(y)-.5 max(y)+.5];
lonlim = [min(x)-.5 max(x)+.5];
```

（3）创建包含 seamount 区域的地图坐标系。

```
worldmap(latlim,lonlim,'none')
```

（4）创建 x 和 y 的 Delaunay 三角形。

```
tri = delaunay(y,x);
```

（5）生成一个组合了三角形和 z 值的三维表面。

```
h = trisurf(tri,y,x,z);
```

（6）通过向 x-y 平面投影，将表面映射到坐标系中。

```
project(h,'yx')
```

（7）给生成的地图中添加一个默认的比例尺。

```
scaleruler
```

结果如图 42-9 所示。

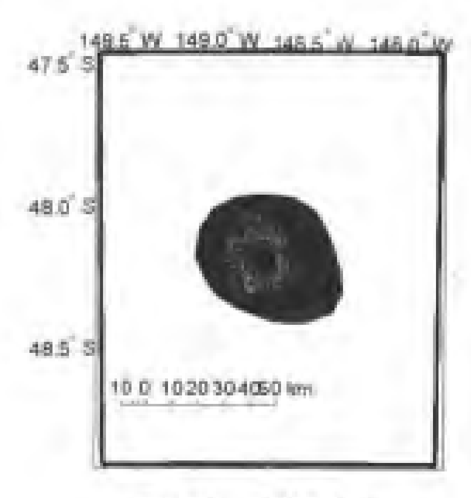


图 42-9 数据三角化

42.4.6 向量图

可以将地图制作工具箱的投影计算结果与 MATLAB 图形函数一起使用，绘制向量地图。此时使用柱面投影是最好的，因为北向上，南向下，东西方向位于正交轴上。

下面的例子在世界地图中叠加一个表面的坡度向量图。表面用 MATLAB 的 peaks 函数生成。

```
figure; axesm mercator; framem; gridm
load coast
plotm(lat,long,'color',[.75 .75 .75])
[u,v] = gradient(peaks(13)/10);
[lat,lon] = meshgrat(-90:15:90,-180:30:180);
[x,y] = mfwdtran(lat,lon);
h = quiver(x,y,u,v,.2,'r');
trimcart(h)
```

生成图 42-10。

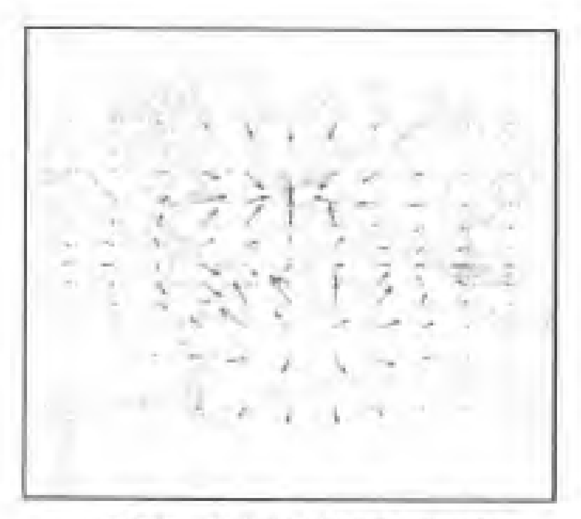


图 42-10 在世界地图上叠加向量图

对于非柱面投影，还需要一个额外的步骤。使用这些投影时，指南针方向会随位置发生变化。要使方向与地图网格的一致，应该旋转向量，使它们对齐。可以用向量转换函数 vfwdtran 实现。下面将上例同样的数据显示在锥面投影中。

```
figure
axesm('lambert','MapLatLimit',[-20 80])
framem; gridm
plotm(lat,long,'color',[.75 .75 .75])
[x,y] = mfwdtran(lat,lon);
thproj = deg2rad(vfwdtran(lat,lon,90*ones(size(lat))));
[th,r] = cart2pol(u,v);
[uproj,vproj] = pol2cart(th+thproj,r);
h = quiver(x,y,uproj,vproj,0,'r') ;
trimcart(h)
```

结果如图 42-11 所示。

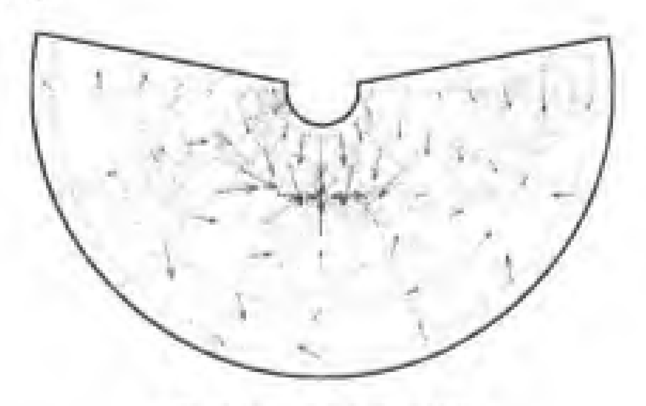


图 42-11　锥面投影的情况

42.5　使用颜色查找表和色条

42.5.1　地形数据的颜色查找表

前面的例子中，用 demcmap 函数给几个数字高程模型（DEM）的地形显示进行了着色。该函数创建了适合渲染 DEM 的颜色查找表。

默认时，这些颜色查找表根据高程或深度数据进行着色。在地形学中，这样的着色称为高程分层设色。

（1）下面载入并打开 korea 数据集中朝鲜半岛的地形数据。

```
clear all; close all; load korea
worldmap(map,maplegend,'meshonly')
```

（2）使用默认的颜色查找表显示朝鲜半岛的 DEM 时，会导致表面特征无法分辨。下面应用默认的 DEM 颜色查找表，并取消图框显示。

```
demcmap(map)
hidem(gca)
```

生成图 42-12。

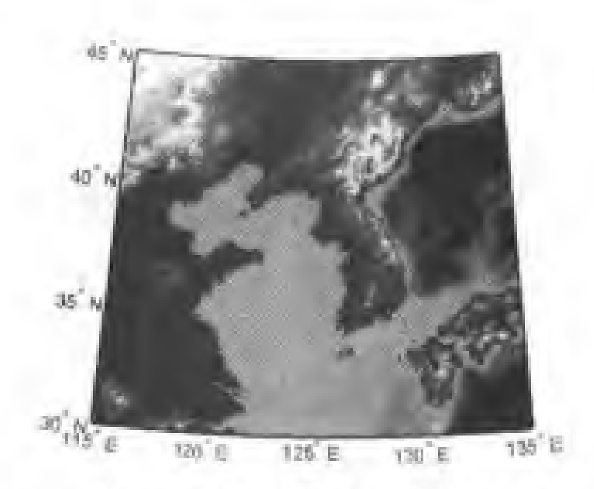


图 42-12 用默认的颜色查找表显示数据

（3）也可以用 demcmap 函数将一定范围内的高程用相同的颜色表示。这将生成拟等值线地图，一个间隔的数据用一种颜色表示。现在用同样的颜色框架进行着色。

```
demcmap('inc',map,500)
colorbar
```

生成图 42-13。注意，demcmap 函数的第 1 个变量值 'inc' 表示第 3 个变量应该解释为值的范围。如果喜欢，可以将第 1 个变量设置为'size'来指定需要的颜色种数。

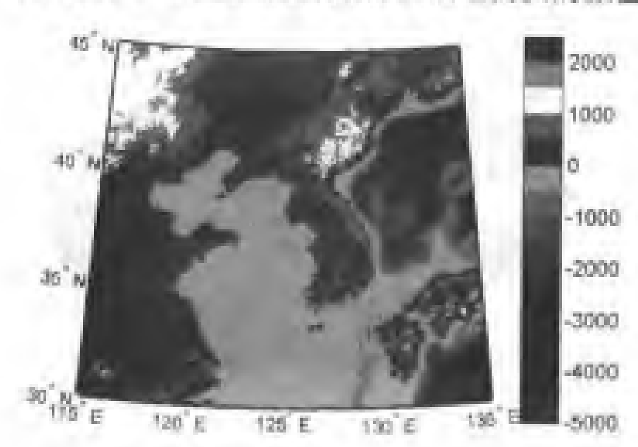


图 42-13 改变着色方式以后的地图显示

42.5.2 等值线颜色查找表

可以创建颜色查找表，使地形数据以外的数据表面也能显示为类似等值线地图的效果。contourcmap 函数可以创建一个颜色查找表，表中颜色按一个固定值增量进行改变。需要的变量是增量值和颜色查找表函数的名称。还可选用 contourcmap 函数来添加和标注色条。

（1）载入世界数据集 geoid，用默认的颜色查找表进行着色。

```
load geoid
figure; worldmap(geoid,geoidlegend)
```

（2）用 contourcmap 函数指定等值线间隔为 10m，在地图下方放置一色条。

```
contourcmap(10,'jet','colorbar','on','location','horizontal')
```

生成图 42-14。

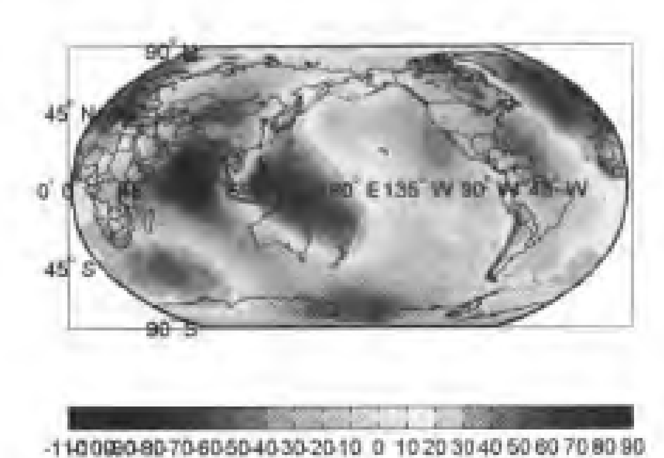


图 42-14　世界地图

（3）如果想对一个范围内的值进行着色，可以给第 1 个变量输入一个均匀间隔的向量。下面指定 5m 间隔，指定低端为 0，高端为 50。

```
contourcmap([0:5:50],...
'jet','colorbar','on','location','horizontal')
```

生成图 42-15。

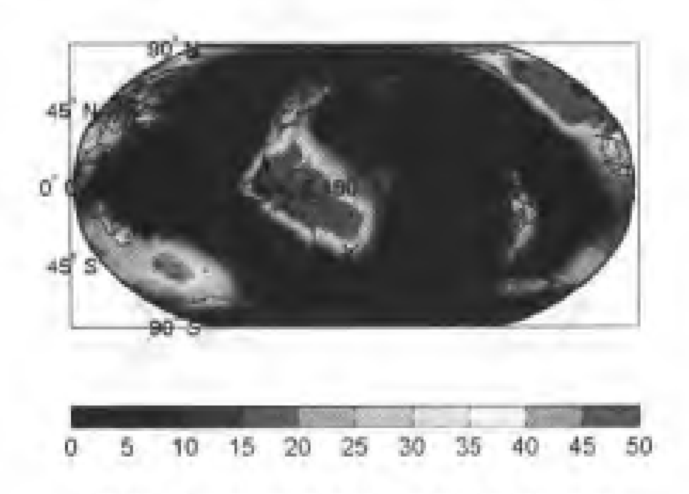


图 42-15　改变颜色范围以后的地图显示效果

42.5.3　政区图的颜色查找表

政区图通过使用对比色，使相邻国家的版图在图上更容易区分开。可以用 polcmap 函数创建这种类型的颜色查找表。该函数创建一个可以在所有色调的颜色中随机选择颜色的查找表。因为颜色是随机的，如果不喜欢生成的结果，可以再次执行 polcmap 函数，生成一个不同的颜色查找表。

（1）为了浏览政区图的颜色查找表，以面片的形式显示 worldlo 数据集。

```
figure; axesm bries
displaym(worldlo('POpatch'))
framem
```

生成图 42-16。该图效果看起来不能另人满意。

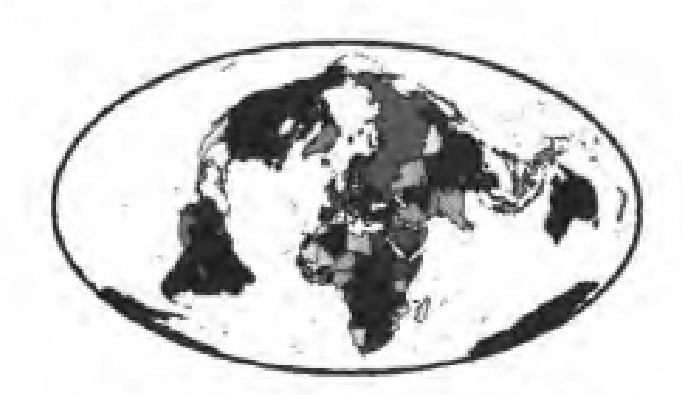


图 42-16 默认设置条件下生成的地图

（2）用 polcmap 函数对面片重新随机着色。

```
polcmap
tightmap loose
```

生成图 42-17。

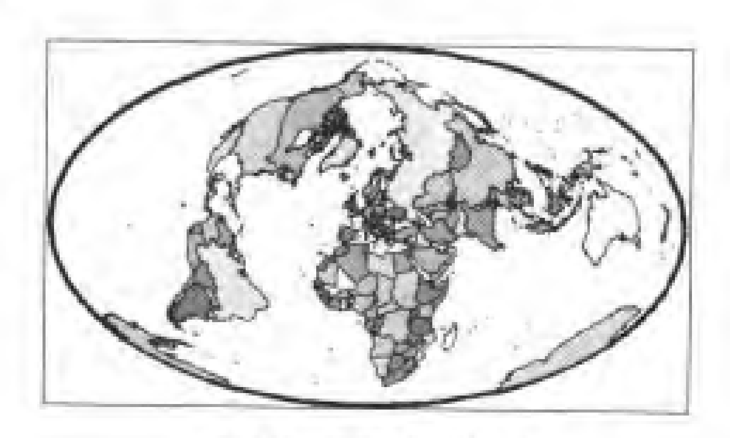


图 42-17 重新着色后的地图

（3）polcmap 函数还可以控制颜色的数目和饱和度。用命令重新指定 256 种颜色和最大的饱和度 0.2。为了保证颜色查找表总是相同的，用变量'state'将 MATLAB 随机数生成函数中的种子重新设置为一个固定值。

```
rand('state',0)
polcmap(256,.2)
```

生成图 42-18。

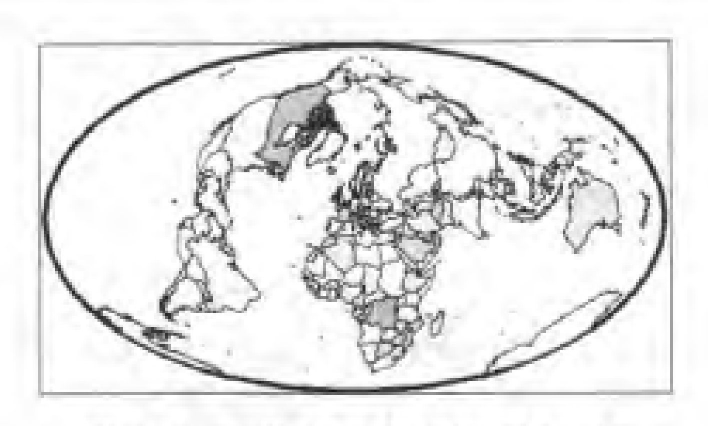


图 42-18　固定随机数种子以后的地图显示效果

（4）为了在最大程度上控制颜色，指定色调、饱和度和亮度的范围。使用与前面一样的随机数颜色索引集。

```
rand('state',0)
polcmap(256,[.2 .5],[.3 .3],[1 1])
```

生成图 42-19。

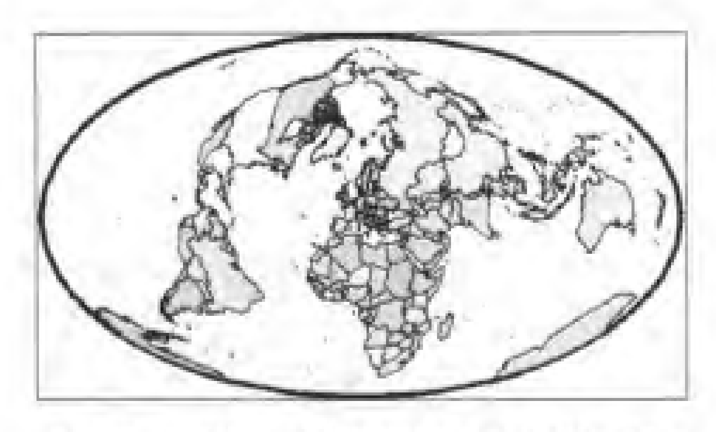


图 42-19　指定色调、饱和度和亮度范围以后的显示效果

42.5.4　标注色条

政区图是定性数据显示的一个实例。许多定性数据集都有与一系列整型值相关的名称，利用 MATLAB 函数 lcolorbar，可以创建一个具有文本标签的色条，这些标签与色条中的颜色相对应。定性色条通常只用于比较小的颜色查找表。

```
figure; colormap(jet(5))
labels = {'apples','oranges','grapes','peaches','melons'};
lcolorbar(labels,'fontweight','bold');
```

生成图 42-20。

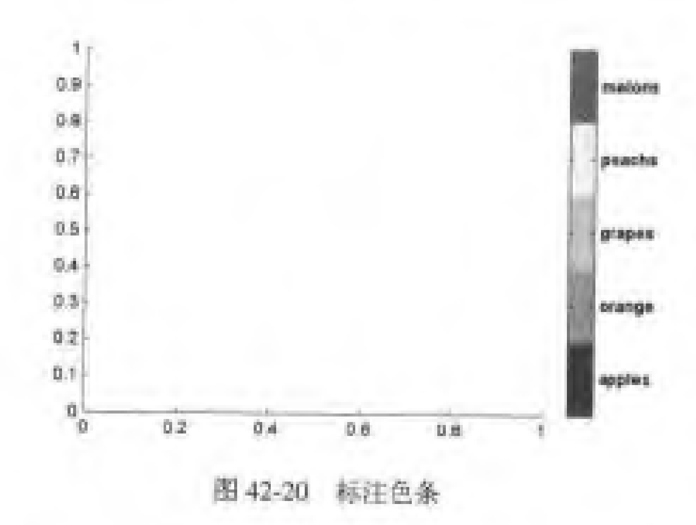


图42-20 标注色条

42.5.5 编辑色条

定性数据的地图常常需要颜色查找表，每个索引值对应于表中的特定颜色。为了避免手工生成这些颜色查找表，可以使用 MATLAB 的 GUI colormapeditor 或地图制作工具箱的 GUI cmapui。利用 cmapui 面板，可以通过在色条上单击对应地实现在颜色查找表中逐个选择颜色入口。为了改变选定颜色的色调和饱和度，可以在颜色轮子上进行拖拉。拖拉红色的滚动条，可以控制 HSV 空间内颜色的亮度。单击“Accept”按钮，返回修改后的颜色查找表。